UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR
DEPARTAMENTO DE INGENIERÍA TELEMÁTICA

INGENIERÍA DE TELECOMUNICACIÓN
SISTEMAS Y REDES DE TELECOMUNICACIONES

PROYECTO FIN DE CARRERA

Implementación del mecanismo de obtención de AS_PATHs disjuntos extremo a extremo en BGP

AUTOR: SERGIO RODRÍGUEZ SÁNCHEZ
TUTOR: RICARDO ROMERAL ORTEGA

27 de enero de 2010

TÍTULO: *Implementación del mecanismo de obtención de AS_PATHs disjuntos extremo a extremo en BGP.*

AUTOR: *Sergio Rodríguez Sánchez*

TUTOR: *Ricardo Romeral Ortega*

La defensa del presente Proyecto Fin de Carrera se realizó el día 27 de enero de 2010; siendo calificada por el siguiente tribunal:

PRESIDENTE: *Isaac Seoane Pujol*

SECRETARIO *Julio Villena Román*

VOCAL *Matilde P. Sánchez Fernández*

Habiendo obtenido la siguiente calificación:

CALIFICACIÓN:

Presidente **Secretario** **Vocal**

"No hay camino que no se acabe
si no se le opone la pereza"

Miguel de Cervantes Saavedra

Agradecimientos

En primer lugar, me gustaría resaltar la importancia que para mí ha tenido la realización de este Proyecto Fin de Carrera. Creo que supone la culminación de un ciclo de mi vida, unos años de duro esfuerzo durante los cuales ha habido muchos momentos buenos, esos 5.0 que tantas alegrías nos han dado, pero también algunos malos que prefiero no recordar.

Son muchas las personas que han hecho posible la consecución de este proyecto. Sin su apoyo, confianza y motivación no hubiera sido factible la obtención de este título.

En primer lugar tengo que agradecer a Ricardo, mi tutor, su paciencia al recibir mis avalanchas de dudas, su continuo apoyo y ánimo durante este tiempo y, sobretodo, las facilidades que me ha brindado durante todo el proyecto. Cada vez que acudía a su despacho en un mar de dudas y agobios, siempre disipaba todas mis vacilaciones y conseguía dar un giro de 180 grados a mi estado anímico.

No encuentro palabras para describir lo que han significado mis compañeros de facultad a lo largo de todos estos años. Para mí han sido más que un apoyo, hemos sabido disfrutar y sufrir juntos lo que no está en los escritos. Recuerdo esas explicaciones magistrales de Virsedita, esos piques con Javi, esas prácticas interminables con Raúl, esas comilonas en casa de Patxi, la ayuda de Jaime que todo lo sabe y la última etapa con Paloma, Joaquín, Grego, Pablo, Isa, Ana y Cris. Tampoco puedo olvidarme de Iñigo, Cristo, Marga, María... y los que me quedarán por nombrar. Gracias a todos por haber estado ahí.

En estos agradecimientos no puedo dejar de incluir a mis amigos, esas personas con las que disfrutar los momentos de ocio que son tan importantes para "desconectar" de los estudios y poder seguir batallando el lunes siguiente. Antonio, Inma, Mandi, Dario, Nacho, María, Nando, Mónica, Ester, Eva y Ruben, entre otros. Muchas gracias a todos por acompañarme durante estos años, por hacerme olvidar por un rato el stress de la uni y por disfrutar de nuestro valioso tiempo juntos.

También les debo agradecer a mis padres, Jesús y Marife, la educación y los valores que me han inculcado desde que tengo uso de razón. Desde que estaba en el instituto y no tenía muy claro mi futuro, ellos y mis hermanos, Alberto y Noelia, me han apoyado en todas las decisiones que he tomado. A la zaga les siguen Sara, Carlos, mis tios y primos que no puedo olvidar mencionar. Gracias a todos por estar siempre ahí.

Como era de esperar, he dejado lo mejor para el final, Patri. Ella ha estado a mi lado durante todos los pasos de este duro camino, apoyándome infinita e incondicionalmente y demostrando una paciencia digna de admirar. Quizás sea la persona que mejor pueda valorar el esfuerzo que ha supuesto para mí esta carrera, todo lo que diga bueno sobre ella se queda corto, ella bien lo sabe. Tampoco puedo olvidarme de su familia: Félix, Tere, Teresa, Paco, los abuelos, Mamen, Juan, etc. Junto a ellos he disfrutado de grandes viajes, momentos inolvidables y hace menos de 2 años, el nacimiento de la sobrina más maravillosa del mundo, Lucía.

Es posible que me haya olvidado de alguien puesto que son muchas las personas que me han apoyado en este largo camino, si es este tu caso, ruego me disculpes y aceptes mis agradecimientos.

Resumen

En la actualidad, BGP es el único protocolo de encaminamiento inter-dominio en Internet. Todo el tráfico entre los diferentes proveedores de servicios de Internet es encaminado a través de BGP. Debido a esta circunstancia, parece una idea bastante atractiva analizar los problemas que pudiera tener este protocolo tan determinante y proponer soluciones que mejoren su rendimiento y prestaciones.

En este proyecto se han presentado las limitaciones de BGP que han dado pie a la solución propuesta en el capítulo 7 de la Tesis Doctoral de Ricardo Romeral[1] y que aquí se ha implementado utilizando un *software* apropiado que se verá más adelante.

En concreto, el desarrollo de este proyecto se centra en la búsqueda de caminos disjuntos[1] interdominio extremo a extremo (AS_PATHs) para su utilización como ruta de respaldo. De esta manera con un sólo camino alternativo se protege completamente el tráfico ante el fallo de cualquier nodo o enlace del camino principal. A la hora de calcular la mejor ruta hacia una determinada red de destino, BGP únicamente realiza un intercambio de información con sus sistemas BGP vecinos hasta obtener una única ruta. Este protocolo, por defecto, no realiza ningún tipo de búsqueda de rutas de respaldo pero si que existen numerosas publicaciones que ofrecen soluciones alternativas para este problema. Una de las primeras soluciones propuestas para este problema fue la de Huang et al. en [2]. La solución propuesta se basa en la utilización de mecanismos de protección independientes en el interior de los dominios y en sus fronteras. Otra solución propuesta por el IETF en [3] consiste en la definición de elementos y procedimientos RSVP para

[1] Caminos que no tienen ningún AS en común, excepto el destino.

la señalización de caminos GMPLS extremo a extremo en distintos tipos de protección.

La solución que se ha implementado en este proyecto consiste en la obtención de una ruta de respaldo (*backup*) mediante la agregación y modificación de algunas funcionalidades del protocolo BGP. Además de realizar las modificaciones indicadas, se comprobará su correcto funcionamiento llevando a cabo la simulación de algunos escenarios significativos y analizando posteriormente los resultados obtenidos en cada uno de ellos. Para concluir este documento se expondrán las conclusiones alcanzadas en base a los resultados de las simulaciones.

Índice general

Lista de Figuras

Lista de Tablas

Lista de Acrónimos

AS	Autonomous System
BGP	Border Gateway Protocol
BOI	Bussiness Oriented Infrastructure
DIT	Departamento de Ingeniería Telemática
eBGP	External Border Gateway Protocol
EGP	External Gateway Protocol
EIGRP	Enchaced Interior Gateway Routing Protocol
GMPLS	Generalized Multiprotocol Label Switching
IBGP	Internal Border Gateway Protocol
IETF	Internet Engineering Task Force
IGP	Internal Gateway Protocol
ISP	Internet Service Provider
LSP	Label Switched Path
MPLS	Multi Protocol Label Switching

NLRI	Network Layer Reachability Information
OSPF	Open Shortest Path First
RIB	Route Information Base
RIP	Routing Information Protocol
RSVP	Resource ReSerVation Protocol
RSVP-TE	Resource ReSerVation Protocolo for Traffic Engineering
SNMP	Simple Network Managing Protocol
UML	User Mode Linux
UPM	Universidad Politécnica de Madrid
VNUML	Virtual Network User Mode Linux

Capítulo 1

INTRODUCCIÓN

En la actualidad, BGP es el único protocolo de publicación de rutas utilizado por las compañías más importantes de ISP en Internet que se utiliza para encaminar todo el tráfico de la red. BGP permite el intercambio de tablas de rutas entre Sistemas Autónomos de diferentes ISP registrados. Este intercambio de información de encaminamiento se hace entre los *peers* externos de cada Sistema Autónomo que deben soportar dicho protocolo. Como se ha mencionado, es el único protocolo utilizado en redes con intención de configurar un EGP(*external gateway protocol*).

El protocolo de enrutamiento fronterizo(BGP) intercambia información de encaminamiento entre Sistemas Autónomos a la vez que garantiza una elección de rutas libres de bucles. BGP-4 es la primera versión que admite encaminamiento entre dominios sin clase(CIDR) y agregado de rutas. A diferencia de los protocolos IGP como RIP, OSPF y EIGRP, no usa métricas como números de saltos, ancho de banda o retardo. En cambio, BGP toma decisiones basándose en políticas de red o reglas que utilizan varios atributos de las rutas BGP transmitidas.

El punto de partida de este proyecto surgió con la realización del estudio tecnológico denominado "*Herramientas de ingeniería de tráfico para redes IP*". En este documento se estudio el funcionamiento de las diferentes tecnologías existentes para gestionar el tráfico de las redes IP. Durante unas 12 semanas se analizó el funcionamiento de MPLS como método para remitir los paquetes de datos mediante el uso de etiquetas, la reserva de recursos en redes de Internet llevada a cabo por RSVP y su aplicación a la ingeniería de tráfico mediante RSVP-TE. También se analizó el comportamiento del protocolo de estado de enlace de pasarela interior OSPF y por último

el protocolo de encaminamiento inter-dominio utilizado para el desarrollo de este proyecto, BGP.

En el estado del arte se centran las explicaciones en el protocolo BGP, que será el protagonista principal de este proyecto, así como las herramientas *software* utilizadas para las simulaciones y posterior procesado de datos: VNUML y quagga.

Tras la realización de este estudio se propuso la posibilidad de implementar una mejora de BGP-4. La idea propuesta fue implementar un mecanismo de obtención de AS_PATHs disjuntos extremo a extremo, ejecutado en segundo plano y que permitiera a cada *peer* de la red obtener dos posibles caminos sin ningún *peer* intermedio en común para alcanzar un mismo destino de la red. Por lo tanto, el objetivo de esta nueva implementación propuesta sería obtener una ruta disjunta alternativa al inicio de la sesión BGP para, en caso de fallo de la ruta principal, poder utilizarla y encaminar todos los paquetes enviados a través de ella y así evitar los retardos provocados por los elevados tiempos de recuperación que poseen las redes que utilizan BGP como protocolo de encaminamiento. Éstas limitaciones se estudian con mayor profundidad en el tercer capítulo de este proyecto.

En primera instancia se pensaban realizar simulaciones con máquinas reales para la utilización de BGP y su posterior modificación, pero tras varias semanas de investigación de las herramientas de simulación existentes (*Network Simulator*, *Opnet*...) se decidió utilizar el virtualizador de redes VNUML que permite una simulación muy próxima a la realidad. La principal complicación de esta herramienta reside en el lenguaje de programación de sus ficheros de escenario, XML, y la necesidad de crear un fichero de configuración diferente para cada *peer* de la red. El aprendizaje de este *software* así como la creación de los ficheros escenario .xml y los de configuración de cada uno de los *peers*(*bgpd.conf*) ha supuesto unas 6 semanas de trabajo. Posteriormente, una vez creado un primer escenario con un funcionamiento correcto, la generación de los nuevos escenarios propuestos en este proyect para el análisis del funcionamiento de BGP fue bastante rápido en comparación con el primero que se acaba de mencionar.

En la página oficial de VNUML[4] se puede obtener gratuitamente un sistema de ficheros para ejecutar cada uno de los *peers* de la red. En la sección de descargas se localizó un sistema de ficheros que incluía en su interior una implementación, en código C, del protocolo BGPv4 que

se desea modificar en este proyecto. Esta implementación de BGP recibe el nombre de *quagga*. Para estas primeras simulaciones del protocolo BGP original se utilizó esta versión de quagga sin ninguna modificación.

Una vez se ha conseguido una simulación correcta del protocolo BGP original el siguiente paso fue descargar de [5] los ficheros fuente(.c y .h) de la versión 0.98.6 de quagga, que es la versión que incluye el sistema de ficheros que se ha descargado para VNUML.

Posteriormente, se procedió a realizar el análisis del código descargado de quagga para estudiar su funcionamiento y estructuras utilizadas en cada uno de los mensajes transmitidos así como las estructuras encargadas de almacenar las diferentes tablas de rutas utilizadas. Esta labor ha sido la más lenta y tediosa del proyecto debido a la escasez o casi inexistencia de comentarios en el código utilizado de quagga. Se tardaron unos 2 meses en analizar y sintetizar el contenido de los más de 500 ficheros que posee esta versión de quagga y otras 6 semanas en modificar y añadir los cambios propuestos en [1] para la realización del nuevo mecanismo de obtención de AS_PATHs disjuntos extremo a extremo. Tras observar las estructuras y funciones utilizadas por quagga, se han tenido que adaptar los cambios para que el código sufriera los mínimos cambios posibles: en lugar de crear un nuevo tipo de mensaje *update* de ruta secundaria se ha utilizado el mensaje *update* existente con un indicador de tipo de ruta(principal o secundaria), en lugar de crear una nueva lista enlazada para almacenar las rutas secundarias disjuntas obtenidas se han introducido estas nuevas rutas en la *Loc-RIB* de cada *peer* con un nuevo indicador de tipo de ruta, etc... Como se puede observar ante estos dos ejemplos, en la mayoría de los casos se ha necesitado crear un indicador de tipo de ruta para diferenciar las acciones realizadas al enviar o recibir cada una de ellas.

Tras realizar las pruebas pertinentes en el escenario más simple, 3 *peers* solamente, y verificar su correcto funcionamiento[1] se procede a generar los nuevos escenarios propuestos en este proyecto y comprobar su correcto funcionamiento en ambos casos(BGP original y modificado) de la misma manera que se realizó con el escenario más simple que se acaba de mencionar.

[1]Se comprueba en cada caso la asignación de rutas: una ruta para BGP original y dos rutas disjuntas para BGP modificado.

En cada una de las simulaciones realizadas se obtendrá el tiempo de ejecución del mecanismo de obtención de rutas completo, se realizarán unas 20 simulaciones de cada caso y posteriormente se calculará la media y varianza de los resultados obtenidos para alcanzar un valor más fiable de la simulación. En cada escenario se han capturado los mensajes *update* intercambiados(mediante *wireshark*), las tablas de prefijos obtenidas al final del mecanismo y el tiempo de ejecución almacenado en los ficheros de registro de cada *peer*.

Una vez obtenidos los resultados numéricos de las 20 simulaciones realizadas para cada uno de los escenarios propuestos en ambos modos de funcionamiento se detallan una serie de conclusiones obtenidas, así como un conjunto de líneas futuras de trabajo que podrían mejorar, todavía más, el rendimiento de este protocolo de encaminamiento.

Capítulo 2

ESTADO DEL ARTE

Para la realización de este proyecto ha sido necesario un análisis exhaustivo del funcionamiento del protocolo BGP (*Border Gateway Protocol*)[6]. De la misma forma se ha llevado a cabo el aprendizaje de nuevas herramientas software que posteriormente fueron utilizadas para la simulación de los escenarios seleccionados, permitiéndonos de esta forma comprobar el correcto funcionamiento del protocolo antes y después de realizar las modificaciones necesarias para el cálculo de rutas de respaldo.

2.1. BGP

2.1.1. Introducción

BGP es un protocolo de enrutamiento entre sistemas autónomos cuya principal función consiste en intercambiar con sus vecinos BGP información sobre la alcanzabilidad de sistemas BGP. Esta información es suficiente para construir un grafo de conectividad entre AS.

Este protocolo utiliza el puerto 179 de TCP para establecer una comunicación entre vecinos. Esto no es lo habitual ya que el resto de protocolos de enrutamiento se ejecutan sobre IP o utilizan UDP. Esto hace posible el envío de mensajes broadcast o multicast para descubrir routers vecinos. Éste descubrimiento no es necesario para BGP, sin embargo, al utilizar TCP evita tener que incorporar un gran número de funcionalidades de nivel de transporte como la fragmentación, secuenciación y retransmisión de datos que ya realiza TCP.

Cuando varios vecinos BGP establecen una sesión TCP, comienzan a intercambiar información BGP en forma de mensajes. Todos los mensajes de este protocolo tienen una cabecera común seguida de los contenidos del mensaje. En la figura 2.1 se puede observar el formato de la misma[7].

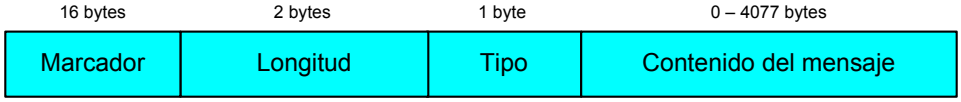


Figura 2.1: Formato de la cabecera de los mensajes.

A continuación se analizará brevemente la función de cada uno de los campos:

- Marcador

  Normalmente es un campo cuyos 16 bytes tienen valor "1" y sirve para comprobar si prevalece la sincronización entre emisor y receptor. Si el receptor encuentra algún valor inesperado en el marcador, significa que ha ocurrido algún error, por lo que enviaría al emisor un mensaje de error y cerraría la conexión.

- Longitud

  Es un campo de 2 bytes que contiene la longitud del mensaje BGP, será como mínimo 19 bytes (longitud de la cabecera sin contenido de mensaje) y como máximo 4096 bytes.

- Tipo

  Campo de 1 byte que nos indica el tipo de mensaje que se enviará a continuación: *open*(1), *update*(2), *notification*(3) o *keepalive*(4).

- Contenido del mensaje

  La longitud de este campo puede oscilar desde los 0 bytes a los 4077 bytes como máximo. Su contenido dependerá del tipo de mensaje enviado.

2.1.2. Tipos de mensaje

Mensaje *Open*

Para iniciar la sesión BGP ambos extremos de la comunicación envían un mensaje *open* inmediatamente después de establecer la conexión TCP. Éste mensaje contiene información importante sobre la configuración y las capacidades de cada uno de los extremos BGP. El formato

de este tipo de mensaje aparece en la figura 2.2 [7].

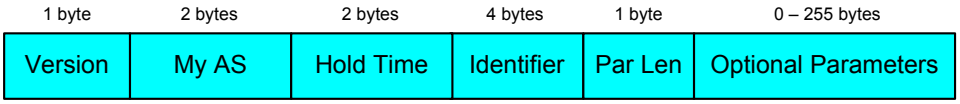


Figura 2.2: Formato del mensaje *Open*.

A continuación se analizará brevemente la función de cada uno de los campos:

- Versión

  Indica la versión de BGP utilizada, actualmente es la 4.

- *My AS*

  Es el número de AS del emisor del mensaje.

- *Hold time*

  Es el tiempo máximo que la sesiçon puede estar en estado *"idle"* antes de cerrarse tras superar el periodo de gracia. Se utiliza el menor *hold time* entre los mensajes recibidos. Su valor mínimo será de 3 segundos y un valor de 0 nos indica que la sesión no se cerrará por eventos de temporización.

- *Identifier*

  Dirección IP del nodo TCP que envía el mensaje. Los *routers* utilizan el mismo identificador para establecer todas sus sesiones BGP.

- *Par len*

  Indica la longitud del campo de parámetros opcionales.

- *Optional parameters*

  Se utiliza para autenticar e incrementar las capacidades como la extensión multiprotocolo y el refresco de rutas.

Si el contenido del mensaje *open* son los enlaces de un *router*, éste enviará hacia atrás mensajes *keepalive* y una copia de la tabla de rutas BGP usando los mensajes *update*. Cuando esta acción sea completada, el *router* envía periódicamente mensajes *keepalive* y *update* incrementales si hubiera modificaciones en la tabla de rutas correspondiente.

Mensaje *Update*

Los mensajes *update* sirven para listar las retiradas e inserción de rutas. Ambas son opcionales, pero un mensaje de este tipo puede retirar rutas, añadir rutas o realizar ambas funciones a la vez. El formato de estos mensajes se muestra en la figura 2.3 [7].

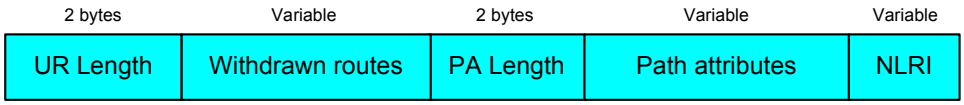


Figura 2.3: Formato del mensaje *Update*.

A continuación se analizará brevemente la función de cada uno de los campos:

- *UR length*

  Indica la longitud del campo de rutas a retirar (*withdrawn routes*), un valor de 0 en este campo indica que no hay rutas que retirar.

- *Withdrawn routes*

  Lista con todas las rutas que se desean retirar, el tamaño de este campo no podrá ser superior al indicado por el *UR length*.

- *PA length*

  Indica la longitud del campo para agregar rutas (*Path attributes*), de la misma forma que el anterior, un valor de 0 indicaría que no hay rutas que añadir.

- *Path attributes*

  Listado de rutas a añadir, funciona de forma similar a las rutas a retirar. Estas nuevas rutas agregadas serán posteriormente utilizadas para seleccionar la mejor ruta cuando un *router* recibe múltiples peticiones para el mismo destino de la red desde diferentes nodos BGP.

  Se va a estudiar la estructura de este campo más a fondo debido a que se trata de uno de los campos que se modificarán en este proyecto para conseguir el cálculo de rutas de respaldo. Este campo puede contener una longitud variable de *Path attributes* pero cada uno de ellos aparece siempre como un trío de valores: <*attribute type*, *attribute length*, *attribute value*>. El campo *attribute type* está formado por 2 octetos. El primer octeto es un *attribute flags*

y el segundo un *attribute type code*. En la figura 2.4 [6] se puede observar su estructura y a continuación explicaremos el significado de sus contenidos:

- Flag 1 (*Optional bit*): Si vale 1 indica que el atributo enviado es opcional.
- Flag 2 (*Transitive bit*): Si vale 1 indica que el atributo es transitivo.
- Flag 3 (*Partial bit*): Si vale 1 el atributo enviado es una parte del total.
- Flag 4 (*Extended Length bit*): Si vale 0 la longitud del atributo ocupa únicamente un byte, pero si vale 1 significa que el tamaño del mismo es superior a 255 octetos y por tanto su longitud ocupará los 2 bytes siguientes al *attribute type*.
- Flags 5 al 8: Estos flags no se utilizan. Por lo que podremos utilizar alguno de ellos para realizar nuestra implementación de rutas de respaldo.

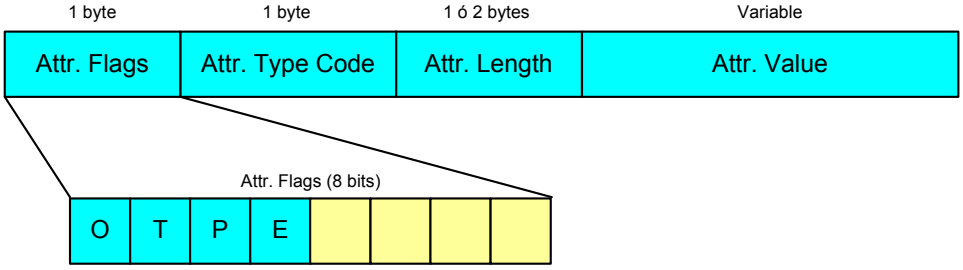


Figura 2.4: Formato de *attribute type*.

El campo *attribute type code* puede tomar diferentes valores en función del contenido del atributo que envíe pudiendo ser: $origin(1)$, $as\_path(2)$, $next\_hop(3)$, $multi\_exit\_disc(4)$, $local\_pref(5)$, $atomic\_aggregate(6)$ y $aggregator(7)$. En este proyecto el campo más relevante será el *as_path* que nos indica la ruta con los AS para alcanzar una red destino determinada.

- *NRLI*

  Contiene los prefijos con el mismo formato que el campo de retirada de rutas. Los prefijos indicados en este campo son aplicados a las nuevas rutas agregadas en el campo de *Path attributes*.

Mensajes de Notificación y *Keepalive*

Un mensaje de notificación se genera cuando tiene lugar una condición de error. Tras enviar este mensaje, el emisor cierra la conexión TCP. Éste mensaje contiene un byte con el código de error, un byte con el subcódigo y datos opcionales.

Los mensajes *keepalive* se envían cuando la conexión se encuentra en estado *"idle"* para asegurarse de que el temporizador no ha expirado. Éste mensaje consiste en una cabecera de mensaje BGP con el campo tipo con valor 4 y sin ningún dato adicional.

2.1.3. Diagrama de estados

En la RFC1771 de BGP[6] aparecen una lista de posibles estados en los que puede encontrarse una sesión BGP siguiendo el diagrama de estados que se muestra en la figura 2.5 [8].

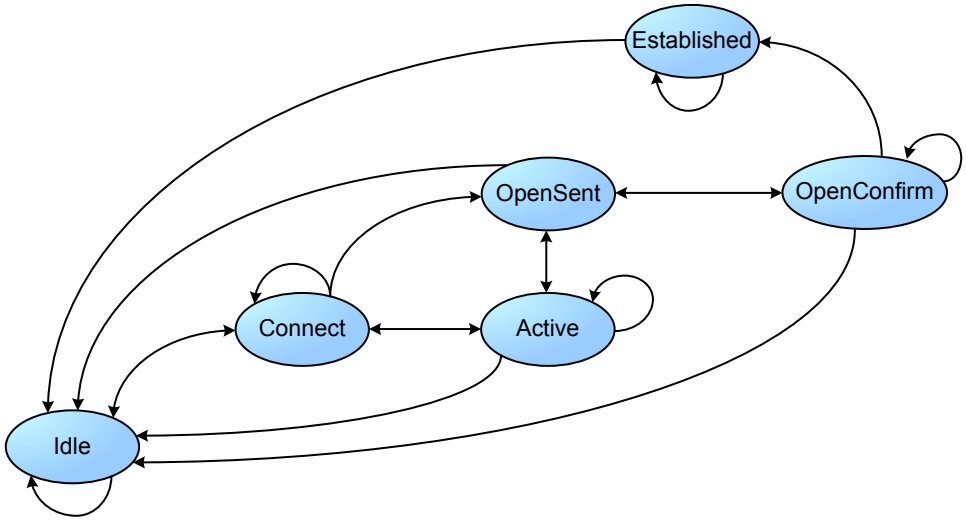


Figura 2.5: Diagrama de estados de BGP.

- *Idle*

  El *router* no intenta establecer una conexión BGP y si un nodo vecino intentara crear una sesión con él, la conexión TCP sería rechazada. Únicamente espera el evento *"start"*.

- *Connect*

  Espera que se complete la conexión TCP y permite otras sesiones entrantes TCP.

- *Active*

  Espera una sesión TCP entrante.

- *OpenSent*

  Envió el mensaje *open* pero no ha recibido todavía respuesta del vecino.

- *OpenConfirm*

  Se ha recibido la confirmación del vecino pero espera la recepción del mensaje inicial *keepalive* para completar el inicio de la sesión BGP.

- *Established*

  Sesión BGP lista, se pueden enviar mensajes *update*, de notificación y *keepalive*.

2.1.4. Manejo de mensajes *update*

La información de encaminamiento que BGP intercambia está basada en la alcanzabilidad de los destinos por cada sistema BGP. Esta información está contenida en el campo *as_path*, dentro del *path attribute* que viaja dentro del mensaje *update*. En definitiva, los mensajes *update* son los portadores del *as_path*, una lista con los AS que estarían "dispuestos" a llevar los paquetes hasta un determinado destino. De esta forma, cada entidad BGP podría generar un subgrafo de la red a partir de la información recibida en los *as_path*. La información necesaria para comprender mejor el mecanismo completo de intercambio de información entre *peers* está definida en el capítulo 9 de la propia RFC de BGP[6] y en el capítulo 7 de la Tesis *"Mecanismos de protección en escenarios IP-MPLS multidominio"*[1].

El mecanismo de selección de rutas y reenvío de las mismas a los *peers* vecinos se realiza en tres fases, en la figura 2.6[1] podemos observar el esquema del mismo.

A continuación se describe con mayor detalle los procesos llevados a cabo en cada una de las fases del mecanismo de selección de rutas:

1. Cuando una entidad BGP recibe un mensaje *update* con una ruta de un *peer* ésta se incluye en la *Adj-RIB-In* que tiene dedicada para el mismo. De esta forma, cada *peer* tendrá, como máximo, tantas *Adj-RIB-In* como vecinos tenga y almacenará en cada una de ellas las rutas aprendidas de ese *peer* en concreto. En esta primera fase del mecanismo de selección de rutas se determina el grado de preferencia de cada una de las rutas recibidas. Cuando hay algún cambio o se recibe una nueva ruta en alguna *Adj-RIB-In* se acciona la fase dos del mecanismo de selección.

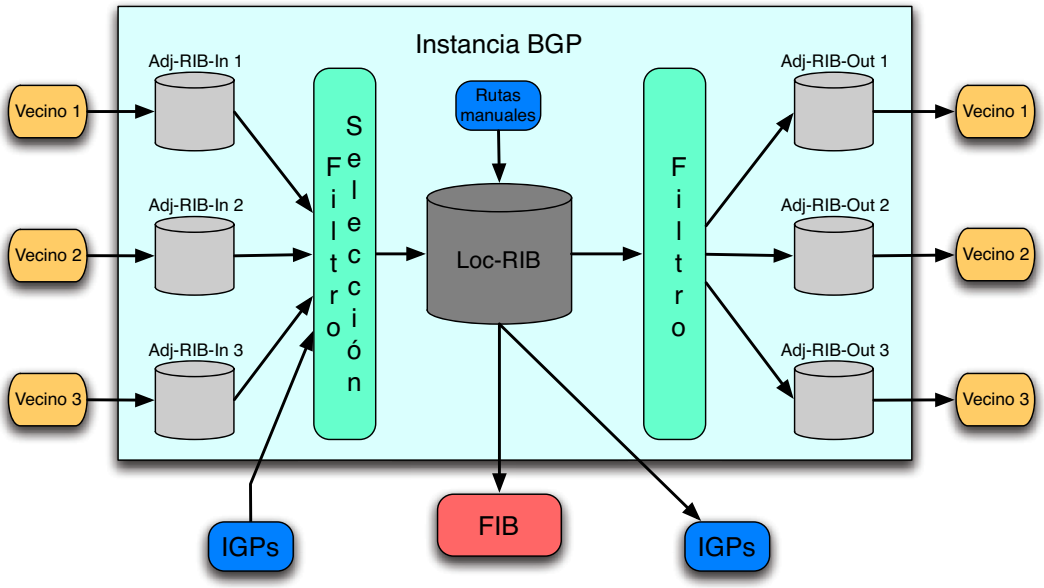


Figura 2.6: Esquema del mecanismo de selección de rutas en BGP.

2. De entre todas las rutas disponibles para un determinado destino, entre todas las *Adj-RIB-In*, se selecciona la considerada mejor (según los criterios y orden de aplicabilidad que aparecen descritos en la RFC[6]). Una vez seleccionada se incluye en la *Loc-RIB*; esta RIB contiene las rutas que éste *peer* utilizará localmente. Cuando se produce algún cambio o alguna nueva ruta es añadida en la *Loc-RIB* se acciona la fase tres del mecanismo de selección.

3. Se seleccionan de entre las rutas de la *Loc-RIB* cuáles se enviarán a qué *peers* BGP; para ello se incluye en las respectivas *Adj-RIB-Out*, la RIB con las rutas que se envían a un determinado *peer* BGP vecino (también tiene tantas *Adj-RIB-Out* como vecinos) y se envían los mensajes *update* correspondientes a los *peers* cuyas *Adj-RIB-Out* se hallan visto modificadas.

En la figura 2.7 aparece un diagrama de flujo en el que se muestra, de manera gráfica, las tres fases mencionadas anteriormente.

Los mensajes *update* son los que se utilizan para intercambiar la información de alcanzabilidad entre los *peers* BGP. Además de esta información hay otros campos dentro de estos mensajes que habría que destacar en este proyecto:

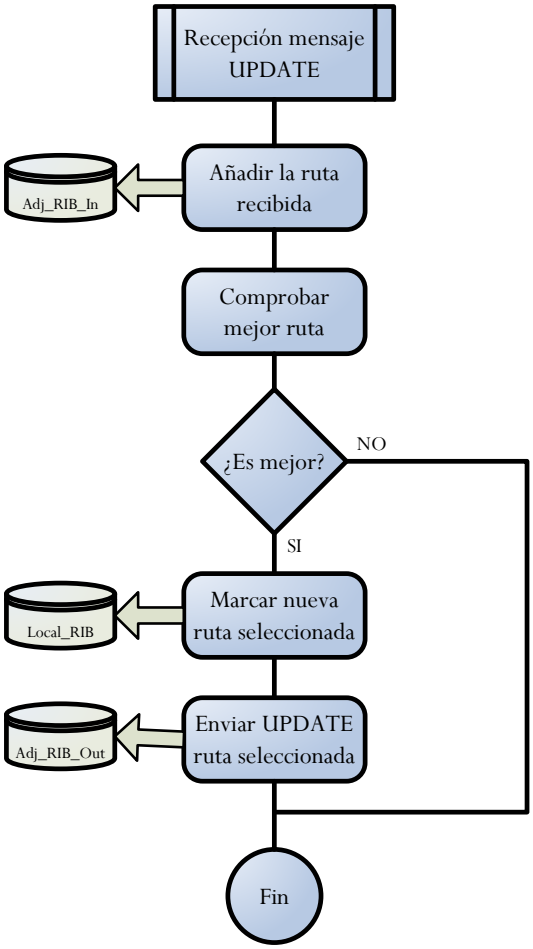


Figura 2.7: Diagrama de flujo para la selección de rutas BGP normal.

- AS_PATH: es la secuencia de ASes que hay que atravesar para llegar a los destinos marcados en el NLRI.
- NLRI: es una lista de prefijos IP a los que se llega por medio de la ruta de ASes que se está anunciando en el mensaje.
- NEXT_HOP: es la dirección IP del router frontera del AS que envía el mensaje *update*. Ésta dirección será utilizada como siguiente salto para los destinos listados en el NLRI.

Los diferentes *routers* frontera de un Sistema Autónomo que utiliza BGP se mantienen informados de las distintas rutas recibidas por medio de iBGP, y de este modo todos los routers BGP del sistema disponen de la misma información. Por ello, excepto que se diga explícitamente lo contrario, cuando nos refiramos a un router BGP de un dominio nos referiremos a cualquiera

de ellos, puesto que todos disponen de la misma información interdominio.

En general, cada *router* BGP recibirá una ruta para un determinado destino de cada uno de sus *peers* vecinos y de todas esas rutas recibidas deberá elegir una de ellas, añadirse como AS en el AS_PATH y anunciarla a sus *peers* vecinos, excepto al *peer* que le envió la ruta seleccionada. También se debe tener en cuenta que en BGP existe una fase de filtrado en la que se establecen ciertas reglas de reenvío mediante las cuales se puede cancelar el envío de rutas hacia determinados *peers* de la red.

Este funcionamiento hace que la información que tiene BGP almacenada en sus RIB no sea suficiente para obtener un grafo de ASes 2-conectado[1] que le permitiese obtener AS_PATHs disjuntos. Más adelante propondremos un conjunto de modificaciones que se han realizado para que sea posible la obtención de ese par de rutas disjuntas.

2.1.5. Ejemplo de propagación de información de alcanzabilidad

En este apartado mostraremos, a través de un ejemplo[1], el mecanismo de selección de rutas de BGP. Se trata de una red de ASes interconectados, formando un grafo 2-conectado. En el ejemplo, se han desarrollado las fases del mecanismo de selección para las rutas destinadas a una red que se encuentre en el dominio AS4, el estado de las diferentes tablas RIB de los *routers* de cada AS y se indicará el envío de cada mensaje *update*.

A continuación se realizarán una serie de aclaraciones sobre la interpretación gráfica del ejemplo desarrollado en la figura 2.8:

- La ruta seleccionada y que por tanto será anunciada a los *peers* vecinos, se marca con fondo azul. Ésta es la ruta que se encuentra en la *Loc-RIB*.
- Si un AS tiene que enviar un mensaje *update* se indica con una flecha de color azul señalando al AS al que se le envía.
- Cuando haya que seleccionar entre varias rutas se elegirá la más corta[2] y si hay empate se escoge una de ellas al azar.

[1] Grafos que tienen al menos dos posibles caminos entre cada par de nodos del mismo.
[2] Se considerará con mayor grado de preferencia la ruta con menor número de AS.

Al desarrollar el ejemplo anterior hemos supuesto que el dominio AS4 acaba de conectarse y ha establecido una sesión BGP con sus vecinos, a los que enviará mensajes *update* indicándoles las redes que está dispuesto a encaminar. Para facilitar la comprensión de este ejemplo teórico, se puede observar que en la figura 2.8 únicamente aparecen representados los mensajes y la información de las RIBs relacionadas cuyo NLRI sea AS4.

En la figura 2.8 (a) AS4 envía los mensajes *update* a sus vecinos (AS1 y AS5). En la figura (b) se puede observar como AS5 y AS1 añaden la nueva ruta recibida en su tabla de encaminamiento (RIB). Como la tabla de ambos se ha visto modificada es necesario ejecutar la segunda fase del mecanismo de selección, elegir la mejor ruta al destino. En este caso como sólo tienen una ruta posible, la seleccionan directamente. Debido a que la nueva ruta seleccionada es diferente a la que había previamente (no había ninguna) ambos AS proceden a enviar la nueva ruta a todos sus vecinos excepto al que se la envió. En la figura (c) vemos como sus vecinos reciben la nueva ruta, la almacenan en sus RIB, seleccionan la mejor y mandan los mensajes *update* a sus vecinos correspondientes.

En la figura (d), se observa que al llegar los últimos mensajes *update*, estos incluyen las nuevas rutas en sus RIBs pero cada uno de ellos comprueba que ninguna de las rutas recibidas es mejor que la anterior, por lo que no hay que actualizar ninguna ruta en la *Loc-RIB* y por tanto ningún AS tiene que enviar mensajes de *update*. Se llega a la estabilidad en la red, teniendo cada AS una ruta para llega al AS4.

Finalmente, todos los dominios de la red tienen una ruta seleccionada en su RIB para alcanzar el destino AS4. Este es el cometido de BGP: proporcionar un camino de ASes para alcanzar un determinado destino.

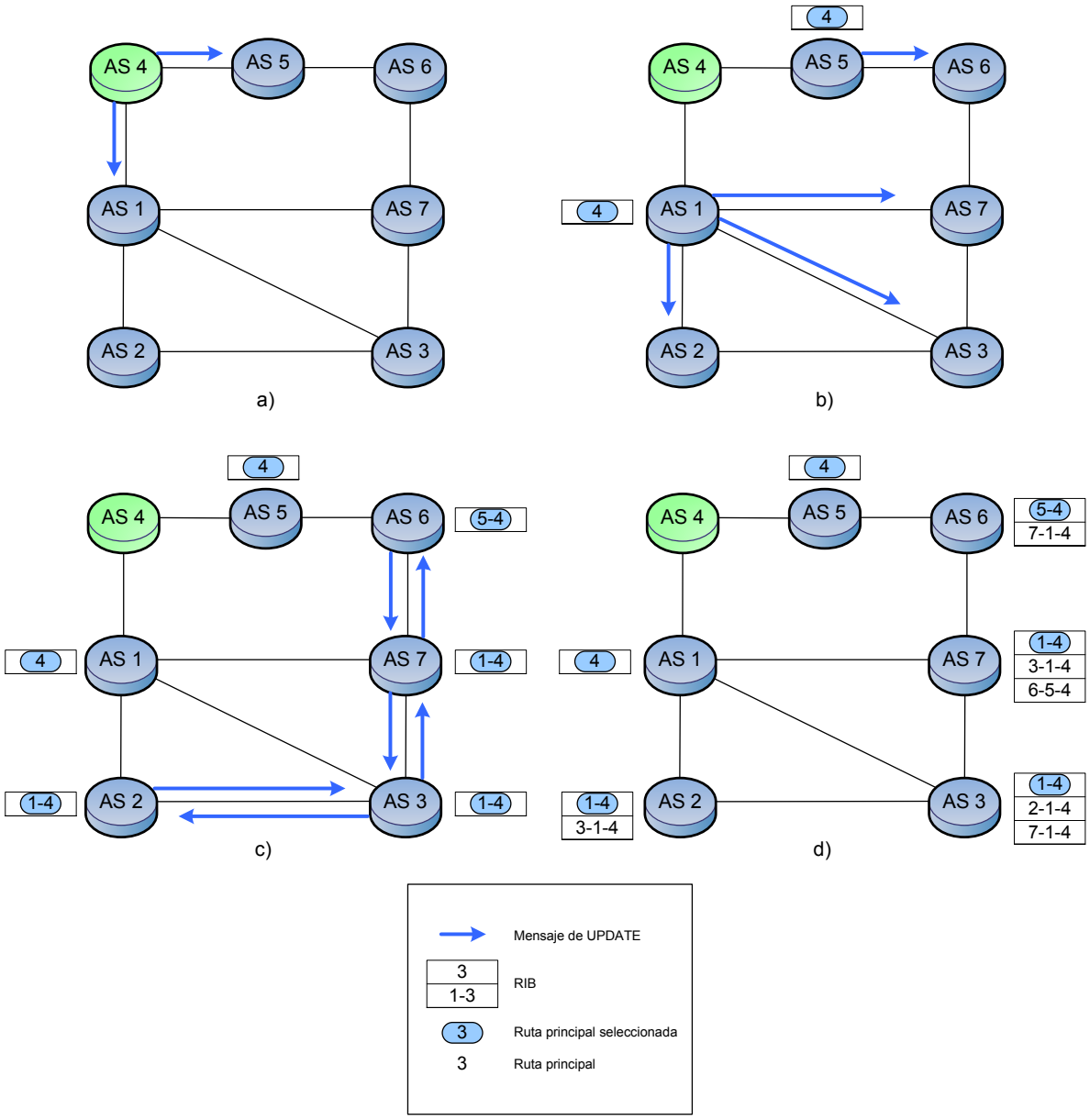


Figura 2.8: Ejemplo de propagación de rutas con BGP.

2.2. QUAGGA

2.2.1. Introducción

Quagga es un paquete de *software* de enrutamiento que provee de servicios de enrutamiento basados en TCP/IP con soporte para protocolos de enrutamiento como RIPv1, RIPv2, RIPng, OSPFv2, OSPFv3, BGPv4 y BGPv4+. Quagga también es compatible con reflectores de rutas BGP y comportamiento de servidor de rutas. Además del tradicional IPv4, quagga también es compatible con los protocolos de enrutamiento con IPv6.

En este software de enrutamiento se utiliza una avanzada arquitectura *software* para poner a disposición de los usuarios las funcionalidades de un potente servidor de enrutamiento con múltiples motores. Quagga presenta una interfaz de usuario interactiva para cada protocolo de enrutamiento, lo que facilitará al usuario la ejecución de simulaciones. En la arquitectura de quagga, se permite al usuario agregar nuevos protocolos o modificar los existentes en su implementación.

La licencia actual de quagga es *GNU, General Public License*, es decir, es un software libre del que cualquier usuario puede conseguir su última versión estable (0.98.6) a través de su página web[9] sin coste alguno.

Quagga es un *software* programado en C[10] que implementa una serie de estructuras y funciones para desarrollar los protocolos mencionados anteriormente. En este proyecto se ha analizado, modificado, comentado y posteriormente probado la parte del código relacionada con el protocolo sobre el que estamos trabajando, BGP. En el capítulo 9 de la documentación oficial de quagga[5] se muestran con mayor detalle las instrucciones necesarias para la creación de los ficheros de configuración de cada uno de los *peer* BGP. Estos ficheros serán necesarios para las posteriores simulaciones que se lleven a cabo. Si desea visualizar un análisis más detallado sobre las instrucciones que se acaban de mencionar puede acudir al apéndice A dónde, a través de un fichero ejemplo, se entenderá fácilmente el significado de cada una de las instrucciones.

2.2.2. Estructuras

Como se ha comentado en el apartado anterior, quagga es un *software* programado en C y que ha sido utilizado en el desarrollo de este proyecto debido a la implementación del protocolo BGP que posee este *software* de enrutamiento.

Aunque quagga consta de múltiples estructuras y funciones que están documentadas en el manual oficial del mismo[5], este proyecto se centrará en los elementos del protocolo BGP de quagga que hayan sido utilizados de manera determinante para la implementación que aquí se ha llevado a cabo.

Como se explica en el capítulo 7 de [1], el mecanismo de selección de rutas de BGP está basado en una serie de RIB que administran la información sobre rutas recibidas, enviadas y seleccionadas. Tal y como se mostraba en la figura 2.6[1] se puede observar que BGP utiliza para cada *peer* tres RIBs que son las siguientes:

- *Adj-RIB-In*: almacena las rutas recibidas por cada vecino, se tienen tantas RIBs de este tipo como vecinos tenga el *peer* y cada una de esas RIBs tendrá tantas entradas como destinos le haya enviado cada vecino.
- *Loc-RIB*: almacena las rutas seleccionadas como mejores, se tiene una única RIB de este tipo con tantas entradas como destinos le hayan notificado al *peer*. En caso de haber recibido varias rutas hacia el mismo destino, esta RIB únicamente guardará la seleccionada como mejor.
- *Adj-RIB-Out*: almacena las rutas que se desean notificar a los vecinos, se tienen tantas RIB de este tipo como vecinos tenga, cada RIB puede tener disponibles tantas rutas como destinos haya en la *Loc-RIB*.

Tras analizar minuciosamente el código de quagga para BGP se ha observado que éste *software* utiliza unas estructuras diferentes a las que se acaban de recordar para la administración de rutas. Esta parte ha sido la más complicada y costosa del proyecto debido a la escasez de comentarios en el código y la inexistencia de documentos oficiales que documentaran la implementación realizada en este *software* de enrutamiento.

En la versión 0.98.6 de quagga se ha observado que en lugar de utilizar un conjunto de RIBs diferentes para cada *peer* de la red, éste realiza toda la administración de las rutas enviadas, recibidas y seleccionadas con 3 listas enlazadas por cada destino de la red. De tal forma que las listas utilizadas para enviar y recibir rutas son similares a las teóricas pero en la *Loc-RIB* se tiene la principal diferenciación:

- *bgp_ adj_ in*: lista enlazada con todas las rutas recibidas por todos los *peers* de la red, en esta lista sólo puede haber una entrada por cada *peer* y destino de la red[3].
- *bgp_ adj_ out*: lista enlazada con todas las rutas que se van a anunciar a cada uno de los *peers*, no posee un límite de entradas por *peer*.
- *bgp_ info*: lista enlazada con todas las rutas recibidas a lo largo de los intercambios de mensajes *update* de BGP. Las estructuras que forman esta lista tienen un atributo "flags" que posee diferentes indicadores entre los que se encuentra `BGP_INFO_SELECTED`, cuando su valor está fijado a 1 significa que esa ruta es la seleccionada como la mejor opción para el *peer* correspondiente.

En la figura 2.9 podemos observar que la implementación de quagga que se ha utilizado posee una estructura *bgp_ node* por cada destino de la red y que a su vez cada *bgp_ node* tiene 3 listas enlazadas para realizar la gestión de rutas.

También podemos observar que cada una de las lista utiliza estructuras diferentes: *bgp_ adj_ in*, *bgp_ adj_ out* y *bgp_ info*. Pero se puede observa en la figura que todas ellas tienen un par de atributos en común:

- *Peer*: es el *peer/router* al que va destinada esa ruta en concreto.
- *As_ path*: es la ruta con la lista de ASes que deberá atravesar el *peer* indicado para alcanzar el destino de esta estructura *bgp_ node*.

Las estructuras que esta versión de quagga utiliza poseen múltiples atributos pero a continuación se presentarán únicamente aquellos que han resultado más relevantes a la hora de realizar esta implementación:

[3] Si un *peer* ya dispone de una entrada en la lista *bgp_ adj_ in*, al recibir una ruta nueva, da por hecho que la ruta anterior debe ser eliminada.

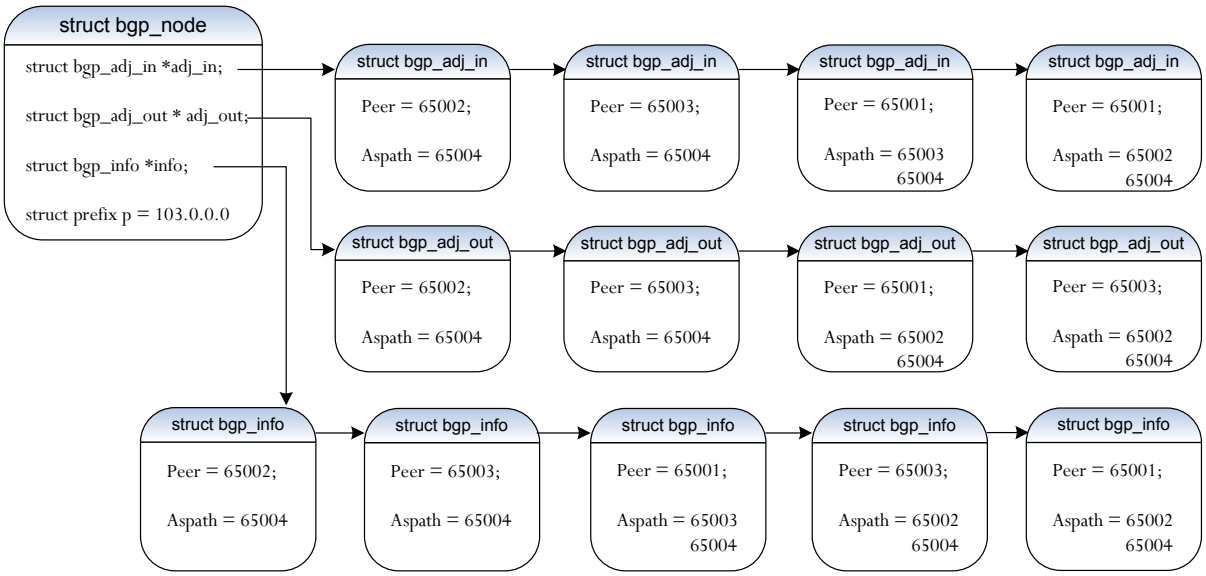


Figura 2.9: Esquema *struct bgp_ node*.

- ***struct bgp_ node***: se tiene un elemento de este tipo de estructura por cada destino de la red y los campos que posee pueden verse en la figura 2.10 y que se explican a continuación:

  - *struct prefix p*: estructura que almacena la dirección IP del destino.
  - *struct bgp_ adj_ in*: puntero a la lista de rutas recibidas que alcanzan este destino.
  - *struct bgp_ adj_ out*: puntero a la lista de rutas listas para ser enviadas con la información para llegar hasta este destino.
  - *struct bgp_ info*: puntero a la lista con todas las rutas recibidas para este destino.
  - *struct bgp_ node*: puntero al siguiente destino de la red[4].

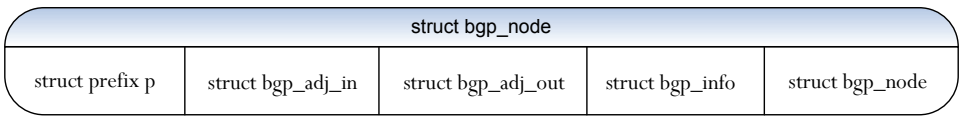


Figura 2.10: Atributos de la estructura *bgp_ node*.

[4]Esto indica que los destinos de la red también están almacenados en una lista de elementos *bgp_ node*.

- ***struct bgp_ adj_ in***: los elementos de este tipo forman listas que parten de una estructura *bgp_ node* y almacenan todos los mensajes recibidos para un determinado destino (sólo puede haber una estructura *bgp_ adj_ in* por cada *peer* de la red). En la figura 2.11 se pueden observar los atributos que hemos utilizado de esta estructura:
  - *struct bgp_ adj_ in *next*: puntero al siguiente elemento de la lista.
  - *struct bgp_ adj_ in *prev*: puntero al elemento anterior de la lista.
  - *struct peer*: estructura que almacena el *peer* que recibe la ruta.
  - *struct attr*: estructura que define los atributos de la ruta recibida y que se verá a continuación.

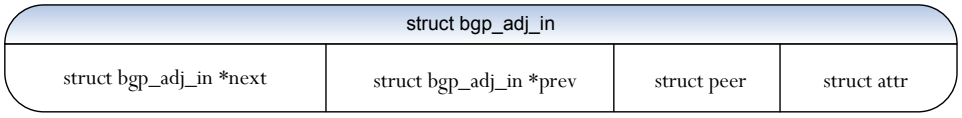


Figura 2.11: Atributos de la estructura *bgp_ adj_ in*.

- ***struct bgp_ adj_ out***: de igual forma, los elementos de este tipo forman listas que parten de una estructura *bgp_ node* y se encargan de almacenar los mensajes listos para ser enviados a un destino concreto. Sus atributos principales aparecen en la figura 2.12 y son los siguientes:
  - *struct bgp_ adj_ out *next*: puntero al siguiente elemento de la lista.
  - *struct bgp_ adj_ out *prev*: puntero al elemento anterior de la lista.
  - *struct peer*: estructura que almacena el *peer* que envía la ruta.
  - *struct attr*: estructura que define los atributos de la ruta enviada y se verá a continuación.

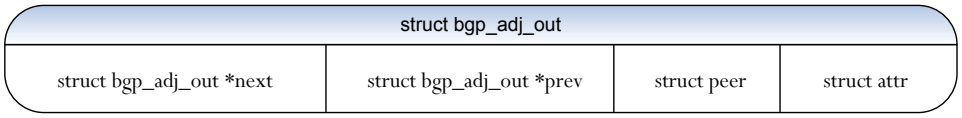


Figura 2.12: Atributos de la estructura *bgp_ adj_ out*.

- ***struct bgp_ info***: los elementos de este tipo también forman listas que contienen todos los mensajes recibidos por un destino determinado y además poseen un indicador con el que "marcan" la mejor ruta seleccionada. Sus atributos aparecen en la figura 2.13 y son los siguientes:

  - *struct bgp_ info *next*: puntero al siguiente elemento de la lista.
  - *struct bgp_ info *prev*: puntero al elemento anterior de la lista.
  - *int flags*: conjunto de bits que por separado indican determinadas situaciones de la ruta a la que pertenecen. En concreto nos interesa conocer el bit `BGP_INFO_SELECTED`, si tiene un valor de 1 indica que esta ruta ha sido seleccionada como la mejor para alcanzar el destino indicado por *bgp_ node*.
  - *struct peer*: estructura que almacena el *peer* que ha recibido la ruta.
  - *struct attr*: estructura que define los atributos de la ruta recibida y se verá a continuación.

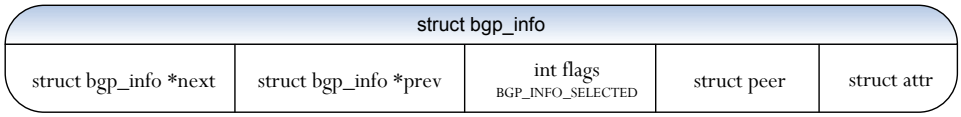


Figura 2.13: Atributos de la estructura *bgp_ info*.

- ***struct attr***: estructura que define los atributos de cada una de las rutas enviadas y recibidas. Esta estructura será la más relevante para este proyecto a la hora de desarrollar la implementación ya que sobre ella residen las principales modificaciones realizadas. En la figura 2.14 podemos observar con más detalle los atributos que posee y que se define a continuación:

  - *struct aspath*: estructura que almacena la ruta de ASes a seguir para alcanzar el destino correspondiente, en ella únicamente cabe destacar dos campos: `int count` que indica el número de ASes que tiene la ruta y `char *str` que es una cadena de caracteres con los ASes de la ruta.
  - *int flag*: campo de 8 bits, cuyos 4 primeros bits se utilizan como flags para diferentes notificaciones y cuyos 4 últimos flags se encuentran sin usar.

- *struct in_ addr next_ hop*: estructura que almacena la dirección IP del siguiente salto de la ruta indicada.
- *int med*(*MULTI_ EXIT_ DISC*): valor utilizado en el proceso de selección de rutas para seleccionar el mejor entre varios vecinos posibles.
- *int local_ pref*: valor utilizado por un *router* BGP para notificar al exterior de su dominio el grado de preferencia sobre una ruta anunciada.
- *int weight*: valor que indica la ruta con mayor grado de preferencia, a mayor valor mejor será la ruta.

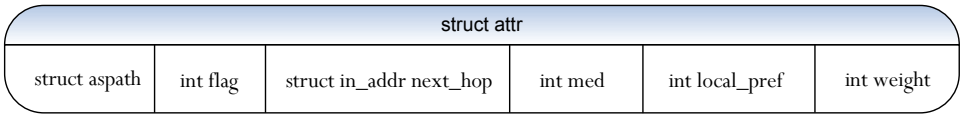


Figura 2.14: Atributos de la estructura *attr*.

Desde el punto de vista teórico estos campos son todos los que se necesitan conocer para más tarde comprender las modificaciones realizadas para llevar a cabo la implementación que tiene lugar en este proyecto.

2.3. VNUML

2.3.1. Introducción

VNUML *(Virtual Network User Mode Linux)* es una herramienta de virtualización libre diseñada para una rápida definición y prueba de escenarios de simulación de redes complejas basadas en el software de virtualización UML (*User Mode Linux)*. VNUML es una herramienta muy útil para la simulación de redes basadas en Linux. Su principal objetivo es ayudar a probar aplicaciones y servicios en redes complejas, pudiendo soportar desde redes de una única máquina Linux a cientos de nodos, sin verse modificada la dificultad necesaria para gestionar la creación de equipos reales.

Desde su desarrollo a finales de 2002, VNUML ha sido ampliamente utilizado en varios ámbitos relacionados con la creación de redes y la informática. Arquitecturas de enrutamiento (como el proyecto IPv6 IST Euro6IX[11]), plataformas de servicios IP multimedia y seguridad lógica son algunas de sus actividades de investigación y desarrollo de aplicaciones. Además, VNUML se utiliza en la educación superior (universidades y colegios técnicos) a fin de construir laboratorios de formación para los estudiantes, en cuyo caso nos encontramos nosotros. En este proyecto se utilizará VNUML para simular desde escenarios simples con 3 nodos a otros algo más complejos que constarán de varias decenas de nodos, en el apartado de implementación se entrará en mayor detalle de estos escenarios.

VNUML es una herramienta que consta de dos componentes principales: XML y su intérprete. XML es el lenguaje de programación utilizado para realizar la descripción de las simulaciones en VNUML y el intérprete del mismo se encarga manejar dicho código y ocultar todos los detalles complejos de UML al usuario.

VNUML ha sido desarrollado originalmente por el Departamento de Ingeniería de Sistemas Telemáticos (DIT) de la Universidad Politécnica de Madrid (UPM) en España. Desde enero de 2008, el desarrollo VNUML está parcialmente financiado por el BOI (*Business Oriented Infrastructure*) gracias a la iniciativa de investigación de la unidad de Telefónica I + D denominada como BSS (*Business Support Systems*).

Este software está liberado bajo *GNU Public License*. Desde julio de 2004, el proyecto está parcialmente alojado en *sourceforge.net*[12].

2.3.2. Funcionamiento

En la actualidad, la virtualización supone una importante alternativa a la implementación de sistemas con equipos reales. En esta breve introducción a VNUML, basada en el trabajo de Fermín Galán[13], se pretende mostrar la potencia y flexibilidad de esta herramienta que simplifica el trabajo de los usuarios automatizando la construcción de escenarios virtuales.

Se podría decir que VNUML permite encapsular una unidad de proceso[5] para su ejecución dentro de un entorno en un equipo anfitrión que emula el entorno real. En nuestro caso el proceso que ejecutaremos será un *kernel* de linux en cuyo interior se habrá instalado previamente el *software* de enrutamiento necesario, quagga, para el desarrollo de este proyecto. En la figura 2.15[13] se puede observar desde la capa física al nivel de aplicación del funcionamiento de VNUML.

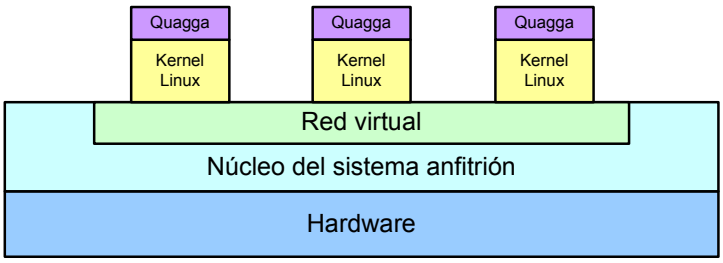


Figura 2.15: Virtualización con VNUML.

[5]Puede ser un programa, un sistema operativo o incluso un equipo completo.

Esto significa que, disponiendo de una máquina lo suficientemente potente que actúe como anfitrión, es posible la ejecución de varias "máquinas virtuales" consiguiendo un funcionamiento igual al que se obtendría realizando ese mismo sistema con máquinas reales.

No hace falta mencionar las numerosas ventajas que implica la utilización de la virtualización de VNUML para el desarrollo de un proyecto. Entre otras se podrían destacar la simplificación de la gestión del sistema y el consiguiente ahorro en costes de infraestructura. Un entorno formado por decenas de equipos reales conllevaría una complejidad de gestión enorme, mientras que utilizando VNUML únicamente actuamos sobre un punto, el equipo anfitrión. Por otro lado, el despliegue de una red con decenas de equipos reales, como sería el escenario 5 simulado en este proyecto, tendría un coste económico y administrativo que hubiera imposibilitado la realización de las pruebas que aquí se han realizado.

Capítulo 3

LIMITACIONES DE BGP

Uno de los principales problemas que presenta BGP como protocolo de enrutamiento de Internet, analizado en detalle en el capítulo 3 de [1], surge a la hora de proteger los recursos del interdominio de la red, es decir, recursos existentes entre varios dominios de diferentes administradores que necesitan cierto grado de protección frente a un error o fallo.

En la solución propuesta por Ricardo Romeral en su Tesis Doctoral, capítulo 7 de [1], se plantea una alternativa sencilla, viable y escalable. Estas serán las tres premisas fundamentales de los argumentos, simulaciones y conclusiones de este proyecto fin de carrera.

Existen dos motivos principales por lo que no es trivial llevar a cabo la protección de recursos interdominio de la red:

- En primer lugar, al utilizarse BGP como protocolo de enrutamiento interdominio, no se dispone de un camino alternativo de ASes, se dispone de información sobre el camino de ASes elegido entre varios ASes, no sobre el grafo de dominios de la red. BGP anuncia una red y una secuencia de ASes (AS_PATH) para alcanzar dicha red destino. Por eso se dice que BGP es un protocolo de *routing* de tipo *path-vector*.

- Por otro lado, al estar involucradas en esta protección redes de gestión administrativa diferente, cada nodo de una red no conoce el grafo interno de otras redes. Debido a esto, un nodo no es capaz de computar un camino alternativo al principal a través de los mismos dominos, el AS_PATH obtenido.

En este proyecto se implementará una solución al primero de los problemas, pero en cualquiera de los casos, hay que tener mucho cuidado con la información compartida, el tiempo de cómputo del conjunto de operaciones a realizar así como la dificultad que conlleva la implantación de la solución a realizar. Como se ha mencionado anteriormente estos factores son cruciales para la validez o nulidad de una posible implementación.

En la mayoría de los estudios que se realizan para mejorar el funcionamiento de cualquier protocolo o servicios de la red es determinante llevar a cabo un análisis sobre el consumo de ancho de banda en el que incurre la implementación propuesta con respecto al consumo original del protocolo, en este caso BGP, e incluso comparando estos valores con otras soluciones propuestas en soluciones alternativas.

La solución que se ha implementado en este proyecto, a partir de [1], responde a la necesidad de calcular una ruta disjunta a la principal para su utilización como ruta de respaldo y así llevar a cabo la protección de los recursos interdominio de la red. Pero para proteger completamente estos datos entre dos nodos de la red que atraviesas varios dominios BGP, existen tres modos de obtener la ruta de respaldo disjunta a la ruta principal.

El primer modo sería computar un par de rutas disjuntas que transcurran por los mismos dominios pero que únicamente tengan en común, a lo largo de todo el camino, el nodo origen y destino. En la figura 3.1[1] se puede observar una red ejemplo de este primer modo de computación de rutas disjuntas.

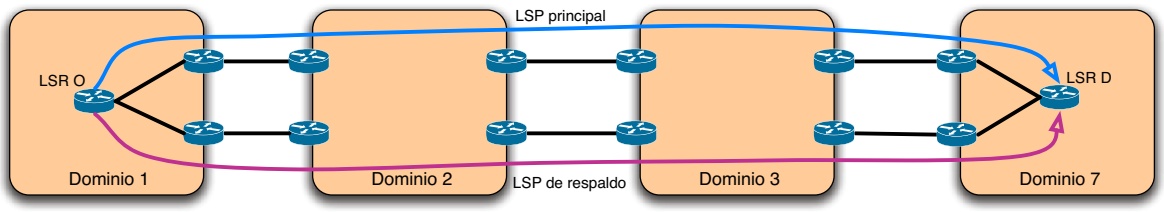


Figura 3.1: Escenario con un único AS-PATH

Un segundo modo es computar cada uno de los caminos por una secuencia de dominios que no tenga ningún dominio en común con el otro, es decir, se tendrían dos caminos independientes. El hecho de tener caminos con dominios disjuntos implica que los caminos también sean disjuntos entre sí. En la figura 3.2[1] se observa una red en la que tenemos dos AS_PATHs disjuntos que transcurren por dominios también disjuntos.

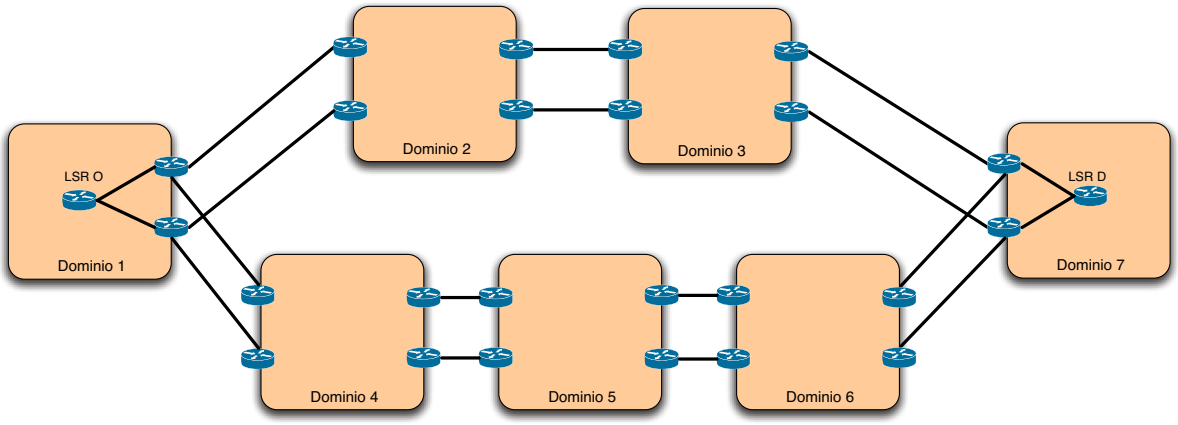


Figura 3.2: Escenario con múltiples AS-PATHs.

Además de estos dos modos presentados, existe un tercer modo que se trata de una solución mixta, en la que parte de las rutas transcurren por distintos ASes, y por tanto son disjuntas en ese tramo, pero por los mismos en otro tramo de la ruta, siendo necesario su procesado para asegurar que sean también disjuntas en ese último tramo por dominios comunes de la ruta y por tanto en la totalidad de la misma.

Cuando se plantean estos tres modos de resolución del problema sobre Sistemas Autónomos, o dominios BGP, no es posible implementar los dos últimos modos con la única información proporcionada por BGP. La agregación de rutas que realiza BGP, el filtrado y selección que lleva a cabo cada uno de los ASes al recibir las rutas y la utilización de los AS_PATH en los anuncios realizados por BGP, hace que no sea posible, en la mayoría de los casos, computar rutas disjuntas como medida de protección de recursos interdominio.

La mayoría de propuestas de esquemas de protección interdominio van destintadas hacia métodos del primer modo de actuación, es decir, computar camino principal y respaldo sobre una secuencia de dominios común.

Sin embargo la solución que se ha implementado en este proyecto, se basa en el cómputo de rutas disjuntas interdominio del segundo modo de actuación mencionado anteriormente. Para ello se supondrá que los dominios son Sistemas Autónomos y que utilizan BGP como protocolo de intercambio de rutas entre ellos. Para llevar a cabo esta nueva faceta del protocolo será necesario agregar y modificar algunas funcionalidades existentes de BGP para lograr el efecto deseado. Estas modificaciones se explicarán con mayor detalle en el siguiente apartado, la implementación.

Capítulo 4

IMPLEMENTACIÓN

Este proyecto consiste en la implementación de una mejora del protocolo BGP para la protección de recursos interdominio en un entorno donde los dominios son Sistemas Autónomos que utilizan BGP como protocolo de encaminamiento. La propuesta realizada en el capítulo 7 de [1] se ha tomado como punto de partida principal para el desarrollo de este proyecto.

En los siguientes apartados se explicará el funcionamiento deseado tras la modificación del protocolo a través de un ejemplo (será la continuación del presentado en el estado del arte para BGP en la figura 2.8) y posteriormente se indicarán las modificaciones necesarias en el código del protocolo BGP implementado por quagga, a modo de resumen, las fases de la implementación han sido:

- Explicación del objetivo del proyecto.
- Ejemplo representativo del funcionamiento deseado.
- Creación de un indicador para conocer el tipo de ruta utilizada (principal o secundaria[1]).
- Creación de una variable estática para indicar el tipo de ruta que se maneja.
- Modificación de los mensajes *update*.
- Modificación del algoritmo de selección de rutas.

[1] Se denomina ruta secundaria a la mejor ruta disjunta de la principal, es decir, a la ruta de respaldo o *back-up* seleccionada

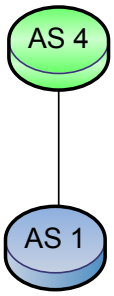
En los siguientes apartados se explicarán con mayor detalle estos puntos, acompañando siempre las explicaciones con gráficos representativos o fragmentos de código que muestren las modificaciones realizadas. Los ficheros completos que han sido desarrollados en este proyecto y que permiten la ejecución de las simulaciones del mismo han sido adjuntados en el *cd-rom* que se presenta junto al mismo, en concreto, pueden encontrar todo el código de quagga en la carpeta `/quagga-0.98.6/` y si les interesa analizar el código del protocolo BGP con las modificaciones implementadas se hallan en `/quagga-0.98.6/bgpd/`.

4.1. Objetivo del proyecto

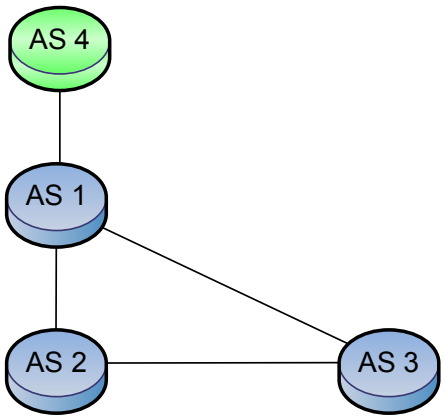
Como ya se ha mencionado en varias ocasiones a lo largo de este documento, el objetivo de este proyecto es realizar una serie de modificaciones sobre la implementación ya existente del protocolo de BGP en quagga para conseguir que dicho protocolo calcule una ruta secundaria disjunta a la principal.

En un primer intento por solucionar este problema se podría pensar en realizar una función para el post-procesado de la información sobre los prefijos de la red de cada *peer*, es decir, utilizar la información que posee cada *peer* para obtener una ruta secundaria disjunta a la principal. El principal problema de esta alternativa es que en la mayoría de los casos la información necesaria no está disponible para todos los *peers*. El motivo de esta falta de información es precisamente el diseño del protocolo de encaminamiento interdominio que se utiliza, BGP, el cual se encarga de filtrar rutas y agregar flujos de información de alcanzabilidad únicamente transmitiendo el mejor camino posible hacia un destino, no todos los caminos posibles.

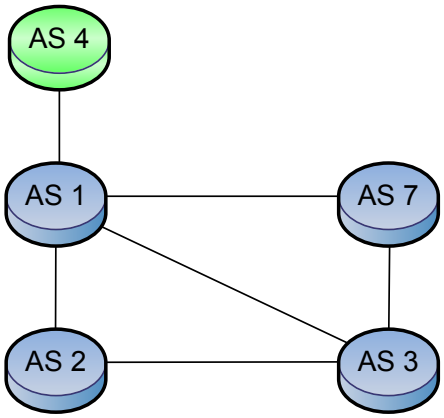
Para comprender mejor esta situación se hará referencia al ejemplo de selección de rutas visto en la página 38 de este documento. Se recuerda que tras intercambiar varios mensajes *update* se alcanzaba un estado final en el que las tablas de prefijos eran estables. Esta situación se observa en el apartado (d) de la figura 2.8. El estado final alcanzado por el protocolo BGP normal presentaba una tabla de prefijos diferente para cada *peer* de la red, por lo que cada uno de ellos tiene una visión diferente de topología de la red para alcanzar un destino. En la figura 4.1 se ve la topología percibida por cada peer. En este caso concreto, los únicos ASes son capaces de obtener una ruta disjunta a partir de la información de la que disponen serían AS6 y AS7.



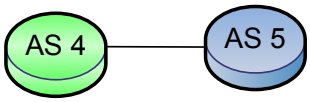
Topología de red percibida por AS1



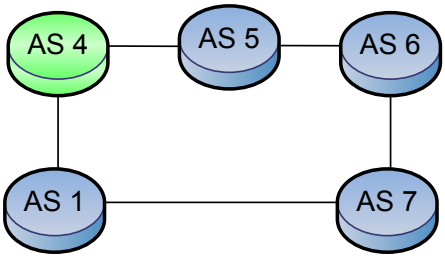
Topología de red percibida por AS2



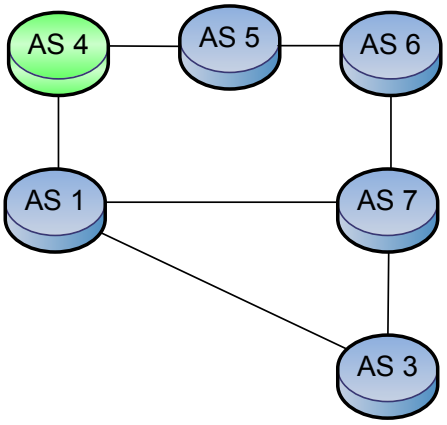
Topología de red percibida por AS3



Topología de red percibida por AS5



Topología de red percibida por AS6



Topología de red percibida por AS7

Figura 4.1: Topologías de red percibidas por cada AS.

4.2. Propuesta de modificación de BGP

Para implementar el nuevo mecanismo de selección de rutas que se viene planteando, sería necesaria la creación de 3 nuevas RIBs: *Adj-RIB-Disj-In*, *Adj-RIB-Disj-Out* y *Loc-RIB-Disj*. Estas tres tablas se encargarían de realizar las siguientes funciones sobre la información de las rutas secundarias:

- *Adj-RIB-Disj-In*: cada *peer* tiene tantas tablas de este tipo como *peers* vecinos tenga, sirve para almacenar las rutas secundarias recibidas por cada vecino para cada destino.
- *Adj-RIB-Disj-Out*: cada *peer* tiene tantas tablas de este tipo como vecinos tenga, sirve para almacenar las rutas secundarias que se deben enviar a cada vecino para cada destino.
- *Loc-RIB-Disj*: cada *peer* tiene una única tabla de este tipo que sirve para almacenar la ruta secundaria seleccionada como mejor para cada destino.

En el desarrollo de este proyecto se ha decidido evitar la creación de nuevas listas enlazadas para poder seguir utilizando las funciones y estructuras definidas por el protocolo BGP de quagga. De tal manera que la solución que finalmente se ha implementado consiste en la adicción de un nuevo indicador en las estructuras de cada una de las listas que indique si la ruta que almacena es principal o secundaria (disjunta).

Debido a las sutiles modificaciones que se realizarán en el código de BGP, se analizará este factor con mayor detalle en los apartados siguientes, el mecanismo de selección de rutas principales no se verá afectado de ninguna forma. Se puede comprobar como se mantienen los tiempos de convergencia actuales de dicho proceso mientras que, en paralelo, se ejecutarán las nuevas funciones que se van a implementar para el cálculo de un ruta secundaria disjunta. Uno de los objetivos que se persigue en este proyecto será la demostración de lo despreciable que es el tiempo de cómputo de ruta secundaria frente al de convergencia del mecanismo de selección en el protocolo BGP normal.

Para realizar el cómputo de rutas secundarias se utilizará un nuevo mecanismo en tres fases, similar al que se utilizó para las rutas principales. En la figura 4.2 se presenta el esquema de

este mecanismo, donde se puede observar que las RIBs contienen rutas principales y secundarias[2].

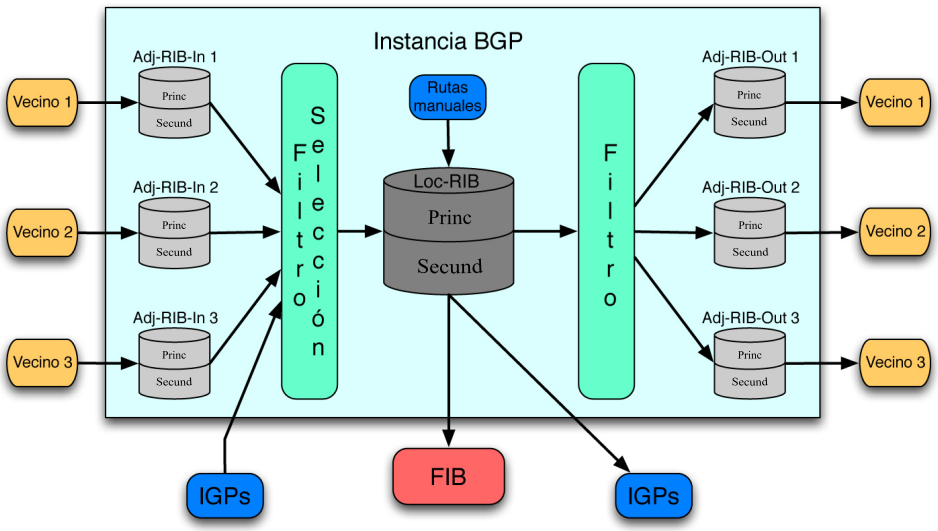


Figura 4.2: Esquema del mecanismo de selección de rutas disjuntas propuesto.

1. Cuando una entidad BGP recibe un mensaje *update* con una ruta secundaria, añade esta nueva ruta en la *Adj-RIB-In*[3] con el indicador de ruta secundaria fijado a 1. En esta primera fase del mecanismo de selección se determina el grado de preferencia de las rutas recibidas y cuando se produce un cambio o se recibe alguna nueva ruta en la *Adj-RIB-In* se acciona la fase dos de este mecanismo. También se activa si se produce algún cambio en la *Loc-RIB*, existe una nueva ruta principal y es necesario un cambio en la ruta secundaria.

2. Para seleccionar la mejor ruta secundaria se deberán analizar todas las rutas contenidas en la *Adj-RIB-In*, tanto principales como secundarias. Si se ha seleccionado alguna se incluye en la *Loc-RIB* con su respectivo indicador de ruta secundaria a 1. Cuando se produce algún cambio o nueva ruta en esta RIB se acciona la fase tres del mecanismo de selección.

3. Se seleccionan de entre las rutas de la *Loc-RIB* cuáles se enviarán a qué *peers* BGP, se incluyen en sus respectivas *Adj-RIB-Out* y se envían los mensajes *update* correspondientes, estos mensajes también deberán llevar un indicador de ruta secundaria.

[2]En esta implementación no se han dividido las RIBs en dos partes, en la figura 4.2 simplemente se pretende representar la diferenciación lograda con el nuevo indicador.

[3]RIB que contiene todas las rutas aprendidas por un *peer* BGP.

Véase que una ruta principal puede ser seleccionada como secundaria pero no al revés. Esto se debe a que el siguiente AS no encaminaría bien los paquetes al no tener obligación de estar utilizando esa ruta secundaria.

Además de este nuevo mecanismo se puede ver en el esquema de la figura 4.3 que el mecanismo para la ruta principal es el mismo, con la única modificación de que, tras obtener la ruta principal y si esta ha cambiado, también activará el mecanismo de selección de ruta secundaria.

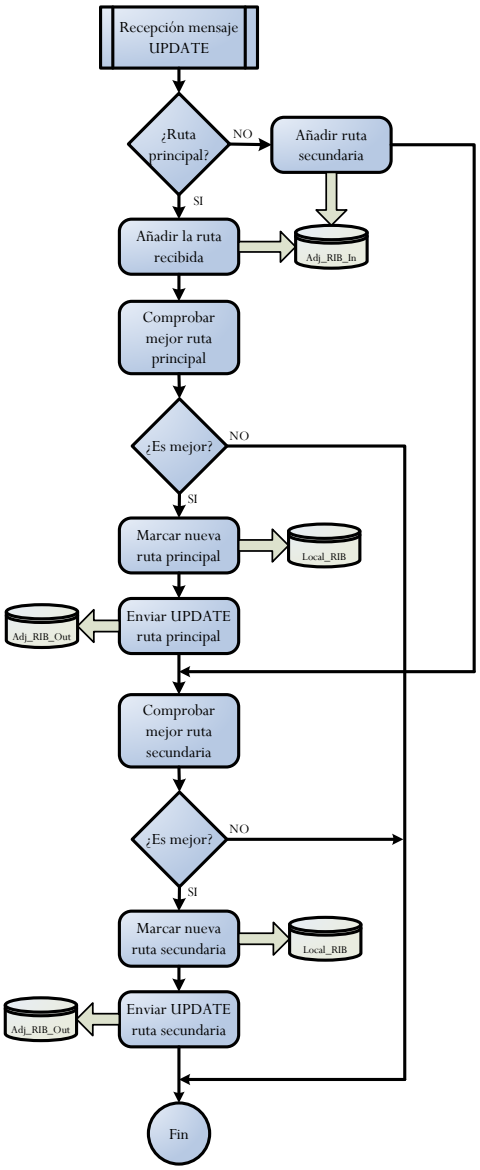


Figura 4.3: Diagrama de flujo para la selección de rutas BGP disjuntas.

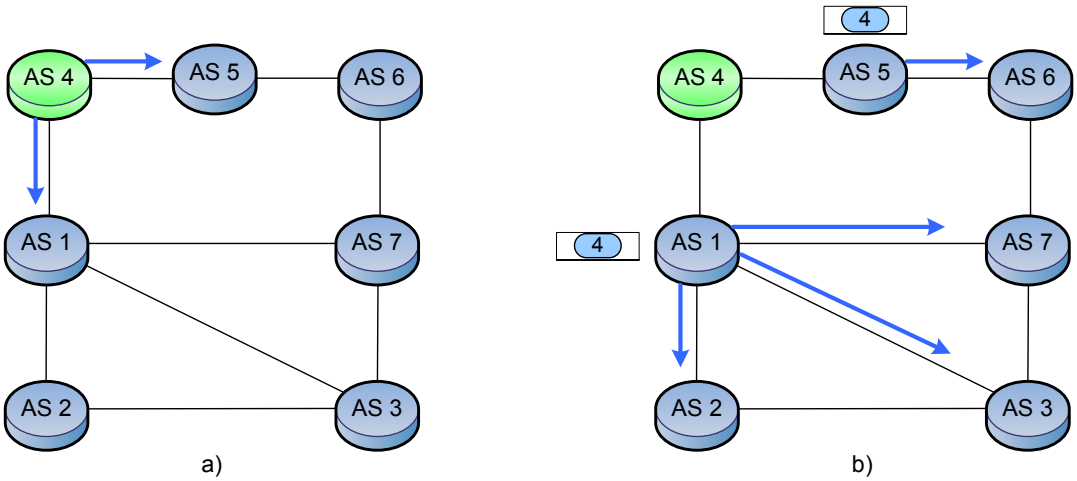
4.3. Ejemplo representativo del funcionamiento deseado

En este apartado se resolverá el mismo ejemplo que se acaba de mencionar en el apartado anterior (véase la figura 2.8). En ese caso se recuerda que BGP únicamente calculaba la mejor ruta principal hacia el destino que se encontraba en el AS4.

Esta vez se utilizará el protocolo BGP modificado tal y como se acaba de explicar por lo que se deberá alcanzar un estado final en el que todos los *peers* de la red tengan en su tabla de prefijos (Loc-RIB) seleccionadas una ruta principal y otra secundaria disjunta. En las simulaciones que se han realizado en este proyecto se verá como existen determinados escenarios que no permiten alcanzar ese estado final deseado a pesar de que el grafo del dominio sea 2-conectado[4].

La figura 4.4 muestra el intercambio de mensajes *update*, tanto de rutas principales como secundarias (flechas azules y verdes respectivamente), que se debería llevar a cabo en el protocolo BGP modificado que se quiere implementar en este proyecto. Finalmente se observa que cuando las tablas de prefijos de los *peers* alcanzan la estabilidad (apartado g de la figura 4.4), cada uno de ellos tiene asignadas la ruta principal y secundaria disjunta.

Antes de pasar a una explicación más detallada del ejemplo desarrollado se debe recordar que en este escenario se está suponiendo únicamente el intercambio de mensajes *update* para obtener las rutas para alcanzar un destino que pertenezca a AS4.



[4]Grafo en el que siempre existen dos caminos disjuntos que unen dos nodos cualquiera.

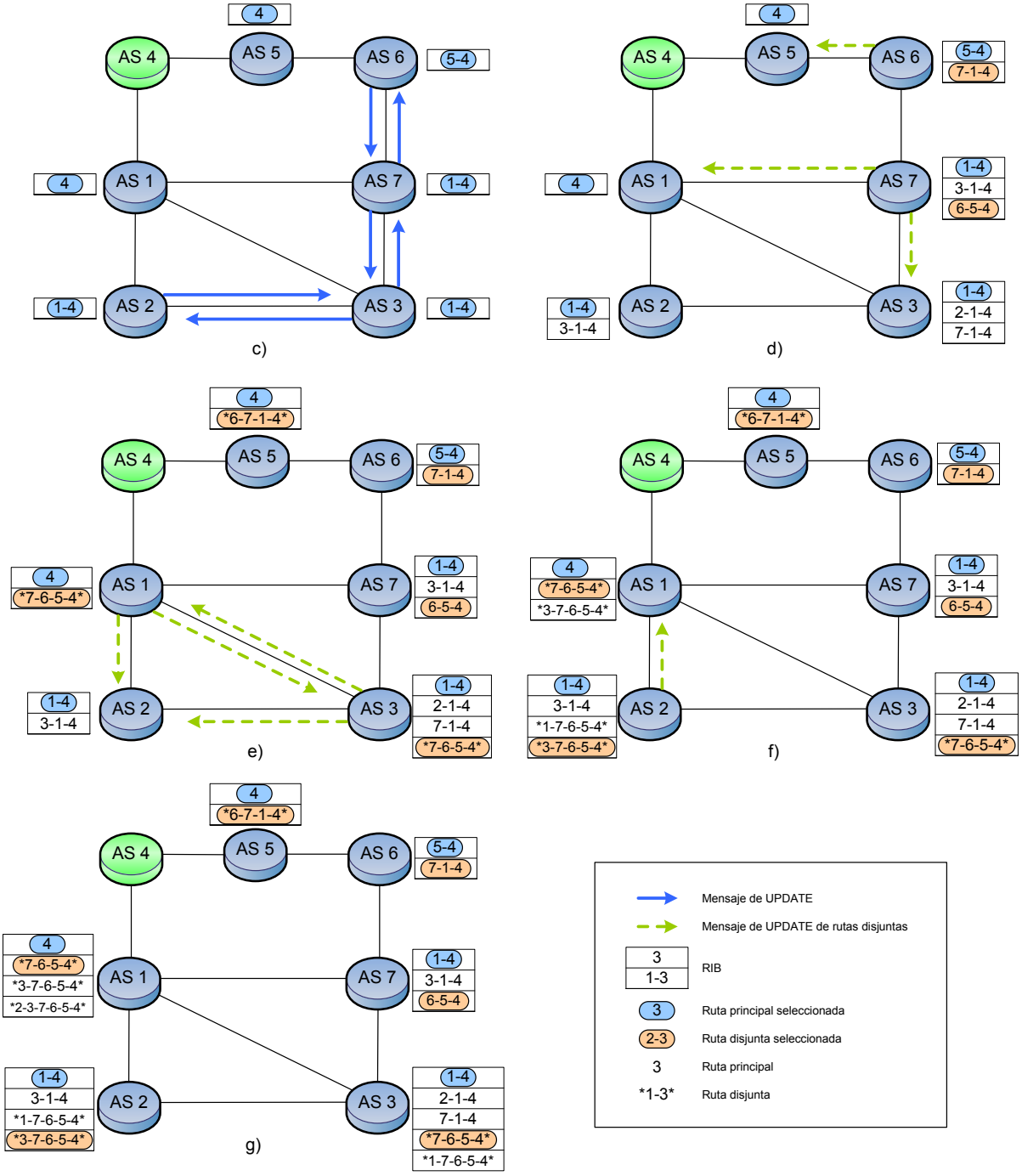


Figura 4.4: Ejemplo de propagación de rutas con BGP.

Como puede observarse en la figura 4.4, los 4 primeros pasos, hasta el d, son los mismos que utilizando el mecanismo de selección de rutas de BGP original. Se observa que en ese paso se han fijado todas las rutas principales de los ASes y ya no se verán modificadas debido a que no se añade ninguna ruta principal nueva y tampoco se modifica el grado de preferencia de las rutas existentes, lo que hace que no se active la fase dos del mecanismo de selección de rutas de BGP.

En el paso d se puede observar que AS6 y AS7 ya poseen en sus listas de prefijos una ruta disjunta a la principal para alcanzar el destino en AS4, por lo que marcarán esa ruta disjunta como secundaria y envían a sus vecinos, excepto al que les envió la ruta, los mensajes *update* de ruta secundaria disjunta. A modo de ejemplo, se presenta la ruta disjunta existente para AS6 que sería 7-1-4 con respecto a la principal 5-4. Se comprueba que ambas rutas comparten el AS final y no tienen ningún AS intermedio en común, es decir, son disjuntas.

Una vez enviados los mensajes *update* de rutas disjuntas, figura e, estas nuevas rutas son almacenadas por cada uno de los vecinos que las reciben en su *Adj-RIB-In* con el indicador correspondiente marcado para identificarlas. Después cada uno de esos AS comprueba su mejor ruta secundaria entre todas las posibles de su lista de prefijos, excepto la que tiene marcada como ruta principal. Todos aquellos AS que hayan seleccionado una ruta diferente a la selección anterior deberán ejecutar el paso tres del mecanismo y enviar a sus respectivos vecinos los mensajes *update* de rutas secundarias correspondientes.

En los apartados f y g de la figura 4.4 se realizan los siguientes intercambios de mensajes *update* referentes a rutas secundarias hasta alcanzar un estado en el que ningún AS tiene ningún mensaje que enviar ya que todas las rutas principales y secundarias de la red han sido establecidas.

En el apartado 4.1 de este proyecto se presentaba su objetivo principal así como un aspecto erróneo sobre el funcionamiento del protocolo BGP en redes 2-conectadas. Se veía como el funcionamiento normal de BGP provocaba una visión errónea de topología de la red para todos los AS de la misma, ninguno de ellos tenía información suficiente en su lista de prefijos para conocer la topología completa de la red. En el ejemplo que se acaba de presentar se puede observar como en el estado final, la visión de la red de AS3 ha cambiado considerablemente con respecto a BGP normal.

En la figura 4.5 se presenta el cambio al que se acaba de hacer referencia. En la parte izquierda de la figura se observa como el AS3 tiene una visión parcial de la topología debido a que desconoce la existencia de rutas através de AS5 y AS6. Sin embargo, al intercambiar la información sobre rutas secundarias, parte derecha de la figura, también le llega información sobre esas rutas por lo que se consigue dar a los Sistemas Autónomos una visión más completa de la topología real de la red. Se dice más completa porque podría darse el caso de tener una red con tres posibles caminos entre dos ASes, en cuyo caso uno de ellos sólo conocería dos rutas para alcanzar al otro. De esta forma una de las rutas quedaría "oculta" para el AS que establece las rutas principal y disjunta.

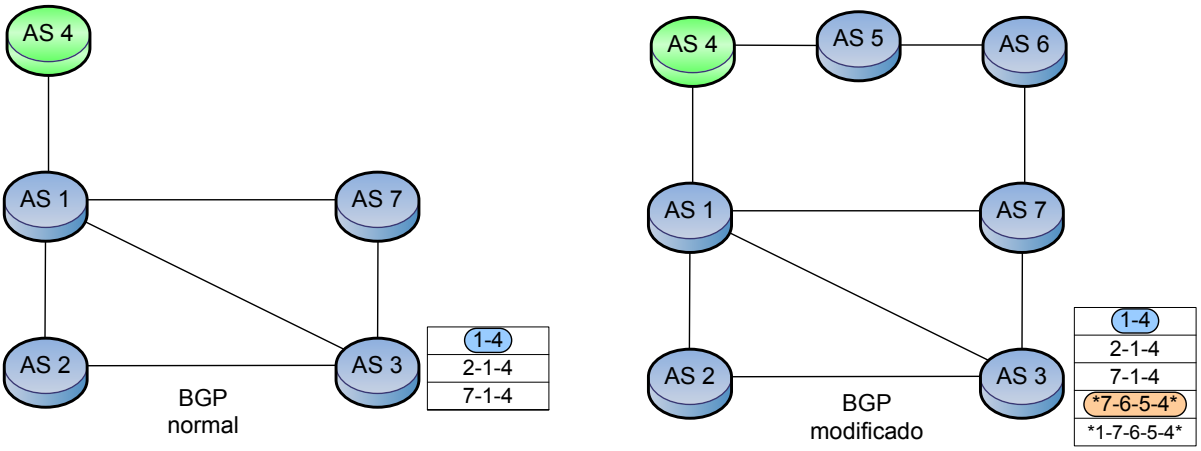


Figura 4.5: Visión de la topología de la red de AS3.

4.4. Modificaciones realizadas sobre quagga

En los apartados anteriores se ha realizado un análisis teórico, incluyendo un ejemplo representativo, sobre el funcionamiento deseado para solventar las limitaciones del protocolo BGP. En los siguientes apartados se detallarán las modificaciones a nivel de código que ha sido necesario realizar para conseguir este funcionamiento.

Normalmente, en la mayoría de los lenguajes de programación, para comprobar el correcto funcionamiento de un nuevo programa implementado se realiza una depuración del código en la propia máquina que lo ejecuta verificando en cada momento el estado de las variables así como el flujo del programa que se está ejecutando. En este caso, la impresión de mensajes por pantalla o la comprobación del estado de las variables implicaría realizar esta acción en cada una de las máquinas virtuales que se están ejecutando. Para facilitar la labor de depuración del código, en esta implementación de código C se ha decidido utilizar los ficheros de registros generados por cada una de las simulaciones en su respectiva máquina virtual. A modo de ejemplo, en el primer escenario de simulación, este fichero recibe el nombre de "escenario1.debug".

Para visualizar el contenido de este fichero en cada momento de la ejecución del programa bastará con abrir su contenido con cualquier editor de texto disponible para poder revisar los mensajes impresos en el mismo. Algunas opciones para la visualización del contenido de este fichero serían:

```
cat /tmp/escenario1.debug
cat /tmp/escenario1.debug | tail
cat /tmp/escenario1.debug | grep "update"
```

A continuación se detallará el funcionamiento de las 3 posibles opciones que se acaban de mostrar. En primer lugar podemos observar que este fichero de registro se encuentra localizado en el directorio `/tmp/` por lo que la primera opción simplemente visualiza su contenido. La segunda opción se puede observar que contiene la opción `tail` cuya misión es mostrar únicamente las últimas líneas del contenido de este fichero. Y por último la opción `grep` permite al usuario buscar las líneas de este fichero que contienen un conjunto de caracteres concreto, en el caso de este proyecto se fijará una palabra "clave" que permitirá únicamente mostrar las líneas de registro que nos interesen para verificar el flujo de la implementación.

Para poder visualizar los mensajes deseados en el fichero de registro "escenarioX.debug" será necesario introducir en el código determinados comandos que escriban estos mensajes en el fichero. A continuación se mostrarán algunos posibles mensajes que se han utilizado en esta implementación:

```
zlog(peer->log, LOG_DEBUG, "Entra: update_main --> Mensaje UPDATE recibido");
zlog(peer->log, LOG_DEBUG, "Entra: update_send --> Mensaje UPDATE enviado");
```

Como se puede observar, el método `zlog` tiene 3 parámetros, el primero indica el objeto tipo log que se encarga del registro, el segundo el tipo de mensaje que se almacena en el registro (LOG_DEBUG) y por último una cadena de carácteres entre comillas que será el mensaje escrito en el fichero indicado anteriormente. En este último parámetro será donde se deberá incluir la palabra "clave" que en este caso ha sido "Entra" para poder filtrar posteriormente los mensajes contenidos en el fichero de registro generado tras la ejecución del demonio bgpd.

Este sistema de notificación de mensajes de registro ha sido utilizada en este proyecto con dos objetivos principales: el primero que se ha indicado anteriormente, para depurar el código y poder desarrollar la implementación que en este caso se quería llevar a cabo. Y por otro lado también ha sido muy útil a la hora de realizar una de las tareas más importantes de esta implementación, el cálculo de tiempos de ejecución. Este sistema ha permitido escribir en el fichero de registros un mensaje en el momento de la recepción del primer mensaje *update* y otro en el momento de recepción del último de ellos, en cada uno de estos mensajes se ha incluído el tiempo, en microsegundos, en el que tiene lugar ese suceso por lo que posteriormente lo único que había que hacer es calcular la diferencia entre ambos tiempos para obtener el tiempo total de ejecución tanto del modo normal como del tiempo obtenido tras la realización de las modificaciones sobre el funcionamiento del protocolo BGP.

En el próximo apartado de este proyecto, "Simulaciones y resultados", se ha incluído una explicación más detallada sobre el método utilizado para la obtención del tiempo total de ejecución del sistema de obtención de rutas disjuntas del nuevo protocolo BGP modificado.

4.4.1. Creación del indicador de ruta de secundaria

En primera instancia al analizar el problema teóricamente y sin haber consultado el código de BGP sobre quagga se planteó la opción de crear nuevas estructuras y funciones para la gestión de las rutas secundarias, pero tras analizar exhaustivamente el código de los ficheros de quagga

que ejecutan BGP se observaron una serie de similitudes en las estructuras de las tres RIBs del sistema (*bgp_ adj_ in*, *bgp_ adj_ out* y *bgp_ info*) que dejaron percibir la factibilidad de la modificación que Ricardo Romeral plantea en el capítulo 7 de su Tesis Doctoral[1]. En este documento se propone el cómputo de rutas secundarias disjuntas únicamente añadiendo un indicador a los elementos de las diferentes RIBs del sistema para indicar si son rutas principales o secundarias. Tras muchas horas de interpretación de código, esta fue la primera modificación que se realizó sobre el código C de quagga.

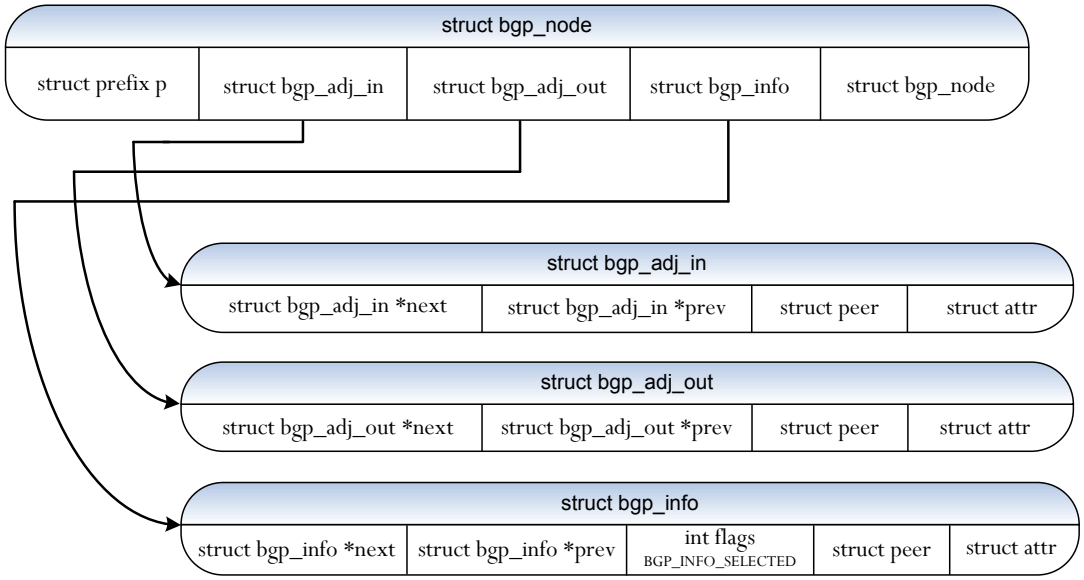


Figura 4.6: Atributos de la estructura *bgp_ node* y sus listas de prefijos.

Para enterder mejor las similitudes a las que se hacen referencia en el párrafo anterior tenemos que remitirnos al estado del arte de este mismo proyecto. En el apartado sobre quagga se exponen brevemente las estructuras que componen las diferentes RIBs de cada Sistema Autónomo. Como se pudo ver en el apartado de estructuras, cada destino de la red posee tres listas enlazadas asociadas para referenciar las diferentes RIBs[5]. En la figura 4.6 se presenta un esquema con las estructuras de *bgp_ node* y las estructuras de las tres RIBs necesarias para cada destino. En esta representación aparecen los campos que se han considerado relevantes para este proyecto, de tal forma que podemos observar claramente las similitudes entre estos tres tipos de estructuras:

[5] *Adj-RIB-in*, *Adj-RIB-Out* y *Loc-RIB*.

- En primer lugar se observa que las tres estructuras poseen un campo *struct peer* que indica la dirección IP, entre otra información, del *peer* vecino de cada ruta.
- En segundo lugar y más importante para el estudio que aquí se está realizando, en todas ellas aparece un campo *struct attr* que como se ha explicado anteriormente representa los atributos de la ruta a la que referencian. Será dentro de esa estructura donde nosotros introduzcamos nuestro atributo de tipo *int* para indicar si se trata de una ruta principal o secundaria.

Como se acaba de explicar, se introduce en la estructura *attr* un nuevo atributo que se denominará *int secundaria*. Este atributo nos indicará el tipo de ruta que contiene cada una de las estructuras. Si este atributo vale 0 se tiene una ruta principal y si vale 1 se tiene una ruta secundaria. En la figura 4.7 se muestra un esquema de la nueva estructura *attr* que se tendría tras añadir este atributo.

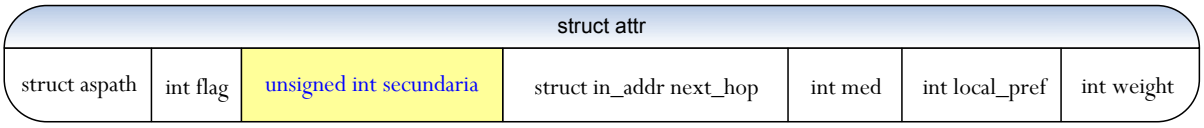


Figura 4.7: Atributos de la nueva estructura *attr*.

La modificación de la estructura *attr* se ha realizado sobre uno de los ficheros adjunto del protocolo BGP sobre quagga: `/quagga-0.98.6/bgpd/bgp_attr.h`, donde se introdujo el comando siguiente: `unsigned int secundaria`. De aquí en adelante, todos los ficheros que se mencionen estarán contenidos dentro de este mismo directorio.

Antes de pasar al siguiente apartado se debe recordar la utilidad de la lista de elementos *bgp_info*. Esta lista sirve para almacenar todas las rutas anunciadas en el sistema, marcando una de ellas para indicar que se trata de la ruta seleccionada. En el caso de BGP normal las rutas sólo podían encontrarse en dos estados: seleccionada o no seleccionada. Pero tras realizar las modificaciones en el protocolo que venimos indicando las rutas podrían tener hasta cinco estados que aparecen desglosados en la tabla 4.1.

Estado	BGP_INFO_SELECTED	BGP_INFO_S_SELECTED	attr->secundaria
Ruta principal no seleccionada	0	0	0
Ruta principal (mejor principal)	1	0	0
Ruta principal (mejor secundaria)	0	1	0
Ruta secundaria no seleccionada	0	0	1
Ruta secundaria (mejor secundaria)	0	1	0

Tabla 4.1: Estados de las rutas en la lista de *bgp_info*.

Como se puede observar en la tabla 4.1, existen 5 estados en los que puede encontrarse una ruta de la lista de elementos *bgp_node* por lo que se ha necesitado la agregación de un nuevo indicador en la definición de uno de los campos de la estructura *bgp_info*. Como se veía en la figura 4.6 esta estructura tiene un campo `int flags` de 8 bits que utiliza para marcar la mejor ruta principal. Como se necesita introducir un nuevo indicador y los 8 bits se este atributo están siendo utilizados por el programa, habrá que incrementar el tamaño del atributo a un entero de 16 bits de la siguiente forma: `u_int16_t flags;`, de esta forma podemos agregar el nuevo indicador, que se denominará `BGP_INFO_S_SELECTED`, para marcar la mejor ruta secundaria y cuya ubicación aparece en la figura 4.8.

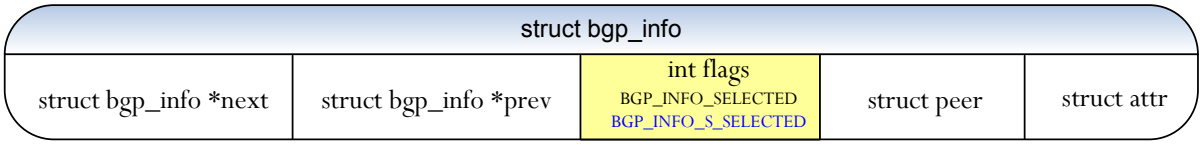


Figura 4.8: Atributos de la nueva estructura *bgp_info*.

Tras introducir este nuevo indicador a los elementos del tipo *bgp_info* se comprenderá mejor el desarrollo realizado en uno de los siguientes apartados sobre la modificación de los mensajes *update* y sus procesos de envío y recepción. Este indicador será crucial para poder discernir cada tipo de rutas y así llevar a cabo una correcta selección de rutas principal y secundaria.

4.4.2. Creación de una variable estática

Al realizar un análisis detallado del código y su funcionamiento se observó que al recibir un paquete de cualquier tipo, el código pasa por numerosas funciones antes de finalizar su labor. En concreto, para los mensajes *update* se analizaron meticulosamente las funciones que utilizaba del fichero `bgp\_route.c`, este análisis se verá con mayor detalle en el siguiente apartado. Finalmente se comprendió la necesidad de crear una variable estática para este fichero que mantenga informados en todo momento a los hilos de ejecución sobre el tipo de ruta con el que trabajan. Se necesita disponer de una variable que sea fijada al recibir un mensaje *update* o en el momento de generarlo para que el resto de funciones por las que pase la ejecución del programa tengan la posibilidad de consultar en todo momento el tipo de ruta que están recibiendo o enviando.

Por los motivos que se acaban de exponer se crea, en el fichero `bgp\_route.c`, una variable estática. Se escribe el siguiente comando al principio del código: `int secundaria = 0;`, con esto se crea una variable estática en C con un valor por defecto de 0 que indica que se comenzará a trabajar con una ruta principal. A lo largo de los siguientes apartados se hará referencia a esta variable para poder desarrollar correctamente las modificaciones necesarias del protocolo.

4.4.3. Modificación del mecanismo de selección de rutas

El mecanismo de selección de rutas se encuentra perfectamente documentado en la sección 9.1 de la RFC del protocolo BGP[6] y en el estado del arte de este proyecto, por lo que en este apartado únicamente se hará referencia a los aspectos relevantes de este mecanismo.

La implementación que se realiza en quagga sobre este mecanismo selecciona la ruta con mayor grado de preferencia entre todas las posibles rutas contenidas en la lista de elementos *bgp_info*. El programa únicamente recorre la lista almacenando la mejor ruta posible, siendo esta la ruta seleccionada.

Sin embargo, en la implementación que aquí se ha desarrollado, en la lista de elementos *bgp_ info* se tienen rutas de dos tipos, principales y secundarias. Esto implica que, en la función `bgp_best_selection` a la hora de buscar la mejor ruta principal o secundaria no se deberán realizar las mismas comparaciones. Las modificaciones realizadas consisten en una serie de condiciones lógicas que comprueban el tipo de ruta recibida en cada ejecución de la función, consultando a la variable estática `int secundaria`. A continuación se presentan los cambios necesarios para cada uno de los procesos de selección en función del valor de la variable estática:

- Si vale 0 se tiene una ruta principal. Como se ha mencionado en la propuesta de modificación de BGP una ruta secundaria no puede ser elegida como principal ya que esta ruta puede no estar siendo utilizada por algún *peer*. Para ello será necesario introducir varias sentencias condicionales (*if, else*) para comprobar que la ruta de la lista que se está comprobando en esa iteración no sea una ruta secundaria.

- Si vale 1 se tiene una ruta secundaria. En este caso podrá seleccionarse cualquier ruta de la lista *bgp_ info* excepto la ruta principal marcada como mejor. Por este motivo, al principio de dicha función se recorre la lista de elementos *bgp_ info* buscando la mejor ruta principal, se almacena su valor que posteriormente se utilizará para compararlo con cada elemento de la lista. Además, una vez se ha encontrado la mejor ruta secundaria, se deberá comprobar que esta ruta es disjunta con la principal seleccionada. Esta comprobación se encarga de realizarla la función `check_disjoint_root` que se analizará con mayor detalle en el último apartado de esta sección.

Al finalizar la ejecución de la función `bgp_best_selection` se devuelven dos variables, una para la antigua ruta seleccionada y otra para la nueva. En función de sus valores llegamos a una conclusión sobre los resultados de la ejecución del mecanismo de selección:

- Si son iguales: significa que la nueva ruta no era mejor que la anterior. Se devuelve un 0 y finaliza la ejecución de funciones.

- Si son diferentes: habrá que desmarcar el indicador de la antigua ruta seleccionada y marcar el de la nueva. En este punto también se realizará una modificación, ya que en función del tipo de ruta que se haya seleccionado se marcará el indicador de ruta principal seleccionada

o ruta secundaria seleccionada, este último indicador ya se explicó en el primer apartado de la implementación de este proyecto.

Con estas modificaciones se consigue el funcionamiento deseado, tal y como se representó en el diagrama de flujo de la figura 4.3 de la propuesta de modificación de BGP.

4.4.4. Modificación de los mensajes *update*

Tras realizar la simulación de varios escenarios simples para comprobar el funcionamiento de BGP se realizó un análisis del código de quagga con el fin de extraer la cadena de funciones que el código de quagga ejecuta al recibir un mensaje *update* y, de esta manera, averiguar los puntos clave del código donde se deberán introducir posteriormente las modificaciones pertinentes para conseguir el funcionamiento deseado.

En primer lugar, al lanza una simulación con el virtualizador VNUML se debe "activar" el demonio BGP de quagga para ponerlo en funcionamiento. Tras lanzarlo, observamos en el fichero `bgp_main.c` la función principal dentro de la cual aparece un bucle infinito (`while(1)`). Este bucle se encarga de inicializar todos los atributos del protocolo BGP y posteriormente de enviar y recibir, métodos `bgp_write` y `bgp_read` del fichero `bgp_packet.c` respectivamente, los mensajes que utilice este protocolo.

A continuación se analizará con mayor profundidad el comportamiento de este código ante la recepción de un mensaje *update*. Los puntos clave de esta cadena de funciones son los siguientes:

- Al recibir un mensaje nuevo se llama a la función `bgp_read` en el fichero `bgp_packet.c`. Esta función comprueba el tipo de mensaje recibido mediante un *case*. Al comprobar que se trata un mensaje de tipo *update* se llama a la función `bgp_update_receive` que se encuentra en el mismo fichero.

- Tras comprobar una serie de condiciones de error en la recepción del mensaje se añade un nuevo evento de mensaje *update* recibido y se incrementa el contador de dichos mensajes para que la función correspondiente quede informada sobre la recepción de este mensaje.

- En el código de quagga existen una serie de funciones que se encargan, recursivamente, de comprobar el estado de las listas de eventos y sus contadores. Una vez se percibe la existen-

cia de un nuevo evento *update* se ejecuta la función `bgp_update` del fichero `bgp_route.c`[6] que posteriormente ejecuta la función `bgp_update_main`.

- La función `bgp_update_main` en primer lugar añade la ruta recibida a la *Adj-RIB-In* utilizando la función `bgp_adj_in_set`. En esta función se ha introducido una modificación ya que en el protocolo BGP original, al recibir un mensaje *update* de un vecino se introducía la nueva ruta sobreescribiendo la anterior[7]. Esto se debe a que cada *peer* únicamente almacenaba una ruta por parte de cada vecino. Sin embargo ahora se necesita almacenar dos rutas por vecino, principal y secundaria disjunta, por este motivo a la hora de introducir la nueva ruta en la *Adj-RIB-In* se debe comprobar si la ruta recibida es del mismo tipo que la existente. Si son del mismo tipo se comprueba que los atributos de la misma hayan sido modificados, en cuyo caso se eliminaría la antigua y se añadiría la nueva.

- Siguiendo en `bgp_update_main` y tras realizar más comprobaciones de condiciones de error, se introduce la nueva ruta en la lista de *struct bgp_ info* donde se almacenan todas las rutas anunciadas de la red, para posteriormente poder calcular la mejor ruta posible.

- Al final de la función `bgp_update_main` aparece la función `bgp_process` que se encargará de calcular la mejor ruta posible y todos los procesos que esto conlleva. También se debe modificar esta parte debido a que el protocolo BGP original únicamente calculaba la mejor ruta posible pero la propuesta de modificación que aquí se propone puede presentar varias opciones acerca de esta ejecución:

  - Si se recibe una ruta secundaria: se fija la variable estática a 1 y se ejecuta la función `bgp_process` una sóla vez.
  - Si se recibe una ruta principal: se ejecuta en primer lugar la función `bgp_process` y después se fija el valor de la variable estática a 1 y se vuelve a ejecutar dicho método para que calcule la mejor ruta secundaria.

  Estas dos posibles ejecuciones del código suponen la realización de una serie de modificaciones sobre la función `bgp_process` para diferenciar su funcionamiento ante una ruta principal o una secundaria, estos cambios se verán a continuación.

[6] Este fichero será donde se desarrolle la mayor parte del trabajo de este proyecto.

[7] BGP considera que cuando un vecino te envía un mensaje *update* con una nueva ruta, implicitamente desea eliminar la ruta anterior.

- La función `bgp_process` ejecuta a su vez `bgp_process_main` que realizará las operaciones restantes. En primer lugar llama a la función `bgp_best_selection` que se encarga de buscar la mejor ruta posible. En el apartado anterior ya se explicó el funcionamiento y las modificaciones requeridas a esta función.

- Si se obtiene una nueva ruta, el siguiente paso de la ejecución será realizar un bucle que recorra a todos los *peers* de la red para comprobar si tiene que notificarles la adjudicación de la nueva ruta que se acaba de realizar. La función que se encarga de realizar estas comprobaciones es `bgp_process_announce_selected`.

- Esta función en primer lugar realiza las comprobaciones pertinentes para asegurarse de que el *peer* que va a analizar se encuentre en el estado establecido y en perfecto funcionamiento. Tras estos procesos, entre los que se encuentra la función `bgp_announce_check`, se comprueba la existencia de una nueva ruta seleccionada.

- En caso de no haberse producido ningún error durante las comprobaciones se procederá a ejecutar la función `bgp_adj_out_set` que se encarga de introducir la nueva ruta seleccionada en la *Adj-RIB-Out* del *peer* que se está analizando para que posteriormente se le envíe su correspondiente mensaje *update*. La única modificación que se ha realizado en esta función es la realización de la comprobación del indicador de ruta principal o secundaria antes de introducir un nuevo elemento en la RIB, ya que si existe un anuncio de ese mismo tipo de ruta para el *peer* que se está analizando habría que sobreescribirlo.

- Si se hubiera producido algún error durante las comprobaciones que se acaban de mencionar, se ejecutaría la función `bgp_adj_out_unset` que eliminará el anuncio hacia el *peer* indicado, enviando un mensaje *withdraw* al *peer* destino. Y si no se tuviera ningún anuncio hacia ese *peer*, se eliminaría su elemento *bgp_ adj_ out* de la lista de la estructura *bgp_ node* indicada por el mismo. En esta función también se ha modificado la condición de búsqueda de elementos en la lista para comprobar el tipo de ruta de cada uno de ellos.

- Tras ejecutar todas las funciones que se han ido explicando, el programa vuelve a la espera de algún evento de mensaje *update* recibido para poner en funcionamiento de nuevo este proceso.

4.4.5. Función de comprobación de rutas disjuntas

En este proyecto se han realizado numerosas modificaciones sobre el código del protocolo BGP de quagga pero únicamente se ha creado una nueva función denominada `check_disjoint_root`. Esta función recibe como parámetro dos cadenas de caracteres con las rutas que se desean comparar y devuelve el resultado obtenido tras comprobar si ambas rutas son disjuntas o no.

El mecanismo de esta función es muy sencillo ya que únicamente realiza la comparación de dos cadenas de caracteres. Pero en este apartado se explicarán de forma resumida los pasos seguidos para realizar dicha comprobación.

Antes de proceder a la explicación se deben conocer algunos aspectos previos:

- El resultado de la comprobación será: 1 si son disjuntas y 0 de no serlo.
- Las cadenas de caracteres de las rutas presentan el siguiente formato: "65002 65003 65004", donde se puede observar que los números representan los Sistemas Autónomos por los que pasa la ruta y los separadores utilizados han sido espacios en blanco.

1. Si alguna de las rutas está vacía se devuelve 1, rutas disjuntas.
2. Se extraen los identificadores de los ASes de la primera ruta recibida utilizando la función `strtok`[8] y almacenandolos en un array.
3. Extraemos de la misma forma los identificadores de ASes de la segunda ruta recibida.
4. Se recorren los elementos del primer array comprobando si alguno de ellos coincide con otro elemento del segundo array, estas comparaciones se realizan con ayuda de dos bucles *for* anidados.
5. Si hubiera algún elemento común en ambas rutas se devolvería 0, las rutas no son disjuntas.
6. Si supera todas las comparaciones sin ninguna ruta común devolvería 1, rutas disjuntas.

[8] Función que se encarga de obtener elementos de una lista donde el separador utilizado se pasa como parámetro.

Capítulo 5

SIMULACIONES Y RESULTADOS

Para comprobar el correcto funcionamiento de las modificaciones que se han realizado sobre el protocolo de enrutamiento interdominio BGP, en los siguientes apartados se presentan algunos escenarios sobre los cuales se ha trabajado y permiten verificar el intercambio de mensajes del protocolo, la asignación local de prefijos en las listas de prefijos de cada uno de los *peers* de la red así como la elección de rutas disjuntas con respecto a la principal.

En los cinco escenarios que aquí se presentan se observa un número distinto de *peers* en cada uno de ellos así como un incremento en la complejidad de las redes. De esta manera se puede probar la influencia de las modificaciones realizadas en este proyecto en función del número de nodos de la red, lo cual parece bastante útil teniendo en cuenta que, en la actualidad, en Internet las redes poseen gran cantidad de ASes.

En los siguientes apartados se presentarán los distintos escenarios mencionados anteriormente. Todos ellos acompañados de una explicación teórica sobre el funcionamiento del protocolo BGP original incluyendo unos gráficos representativos del mecanismo para cada uno de los casos. Posteriormente se mostrará el desarrollo del protocolo BGP modificado también acompañado de gráficos que representarán el intercambio de mensajes *update* y el estado de las RIBs de cada *peer* de la red en cada instante de la ejecución. Finalmente se mostrarán las conclusiones sobre los resultados obtenidos en cada escenario y las mejoras que esta solución ofrece con respecto al protocolo BGP original.

Como se ha mencionado en el apartado anterior, en cada una de las simulaciones se debe obtener el tiempo total de ejecución de la misma. Este tiempo transcurre desde que tiene lugar el primer envío de mensaje *update* de toda la simulación y como finalización el instante de recepción del último mensaje *update* en la misma. De esta forma se obtiene el tiempo total de cálculo de rutas, ya sea principal únicamente o principal y secundaria disjunta. En la figura 5.1 se puede observar para un ejemplo sencillo, red de 4 nodos, el momento preciso al que se ha hecho referencia como instante inicial y final de los tiempos de cómputo de las simulaciones desarrolladas.

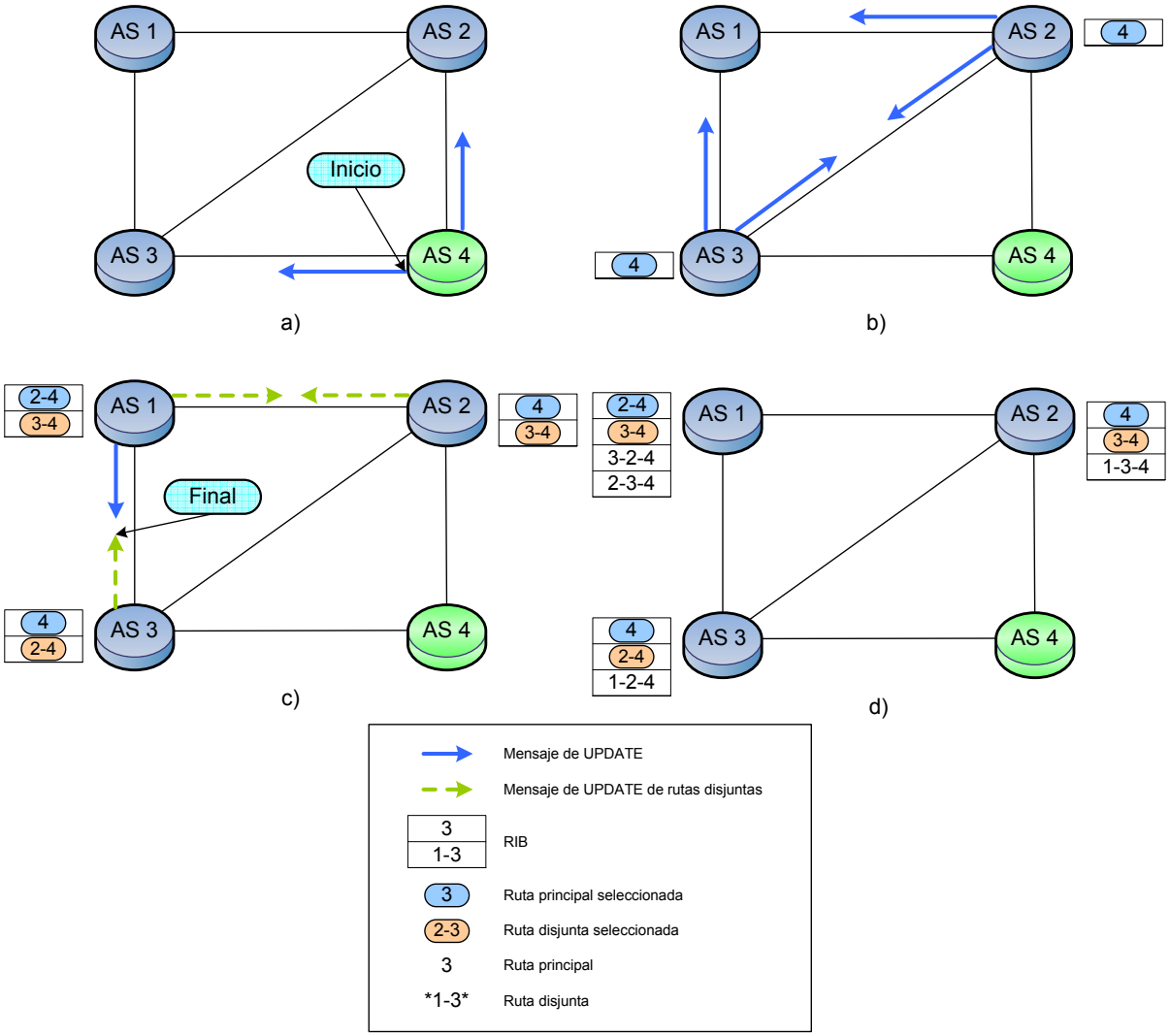


Figura 5.1: Ejemplo de tiempo de ejecución.

En la figura 5.1 se ha podido observar como el primer mensaje *update* enviado ha sido, en esta ejecución en concreto, el enviado por AS4 a AS3 (ap. a) y el último mensaje *update* recibido el que va desde AS3 a AS1 (ap. c). En las siguientes simulaciones se observará que el primer y último mensaje enviado no tiene porqué ser siempre el mismo de manera que se almacenarán todos los tiempos de envío y recepción de mensajes *update* y posteriormente se hallará la diferencia entre el tiempo de envío más temprano y el tiempo de recepción más tardío.

5.1. ESCENARIO 1

Teniendo en cuenta que en este proyecto se está trabajando sobre el protocolo BGP y el objetivo de esta implementación es conseguir que dicho protocolo compute dos rutas, una principal y otra secundaria disjunta a la primera, parece lógico que el número mínimo de *peers* que se puedan simular en un escenario de pruebas sea 3. De esta forma existe la posibilidad, siempre y cuando el grafo de la red sea 2-conectado, de establecer ambas rutas hacia un destino de la red.

En este primer escenario se presenta el escenario más simple que se puede simular para la comprobación del funcionamiento de las modificaciones realizadas sobre el mecanismo de intercambio de mensajes y selección de rutas del protocolo BGP. La red de dicho escenario aparece en la figura 5.2. El Sistema Autónomo para el cual van destinadas las rutas que se están intercambiando aparece en verde y el resto de ASes en azul.

Para realizar un análisis más adecuado sobre las mejoras producidas por la implementación que se ha realizado en este proyecto se llevará a cabo la simulación de este escenario utilizando el protocolo BGP original y posteriormente utilizando el nuevo protocolo BGP que se ha modificado. De esta forma se verán los cambios producidos en las tablas de prefijos de cada *peer* así como los tiempos de ejecución de cada una de las simulaciones.

5.1.1. Funcionamiento

En primer lugar se analiza el proceso que llevaría a cabo el protocolo BGP original mediante unos gráficos representativos que se pueden observar en la figura 5.3 y se explicarán con mayor detalle posteriormente.

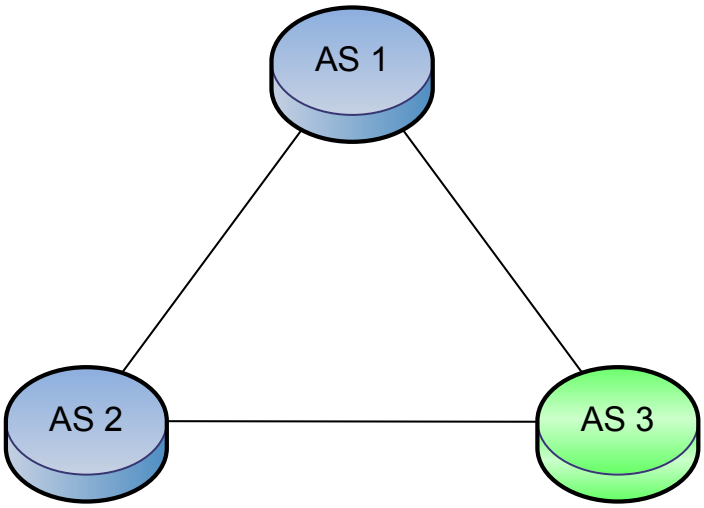


Figura 5.2: Escenario 1: Topología de la red.

Debido a la simplicidad de este primer escenario, no será necesaria ninguna aclaración adicional. Únicamente observar como AS3 comienza enviando el mensaje *update* y posteriormente AS1 y AS2 hacen lo propio al recibirlo. De esta manera notifican a sus vecinos la agregación de una nueva ruta a su *Loc-RIB*. Se puede observar que tanto AS1 como AS2 no envían a AS3 la nueva ruta aprendida ya que ésta procede de él mismo. Finalmente se alcanza el estado final donde todos los *peers* de la red tienen asignada una ruta para alcanzar el destino en AS3.

En la figura 5.4 se puede observar el intercambio de mensajes producido y el estado de las RIBs de los diferentes *peers* de la red utilizando el protocolo BGP modificado en este proyecto.

Se puede observar como el intercambio de mensajes es exactamente el mismo que el que se producía utilizando el protocolo BGP original pero, en este caso, se puede ver como el estado final de las RIBs de los diferentes *peers* posee dos rutas seleccionadas en lugar de una. Por un lado aparece la ruta principal, marcada en azul, y por otro la ruta secundaria, marcada en color salmón. Para este escenario no ha sido necesario enviar ningún mensaje *update* de ruta secundaria puesto que, por ejemplo, para AS1, al asignar la nueva ruta no puede enviarsela a AS3 porque es el AS destino y tampoco a AS2 porque está dentro del AS_PATH de la nueva ruta establecida.

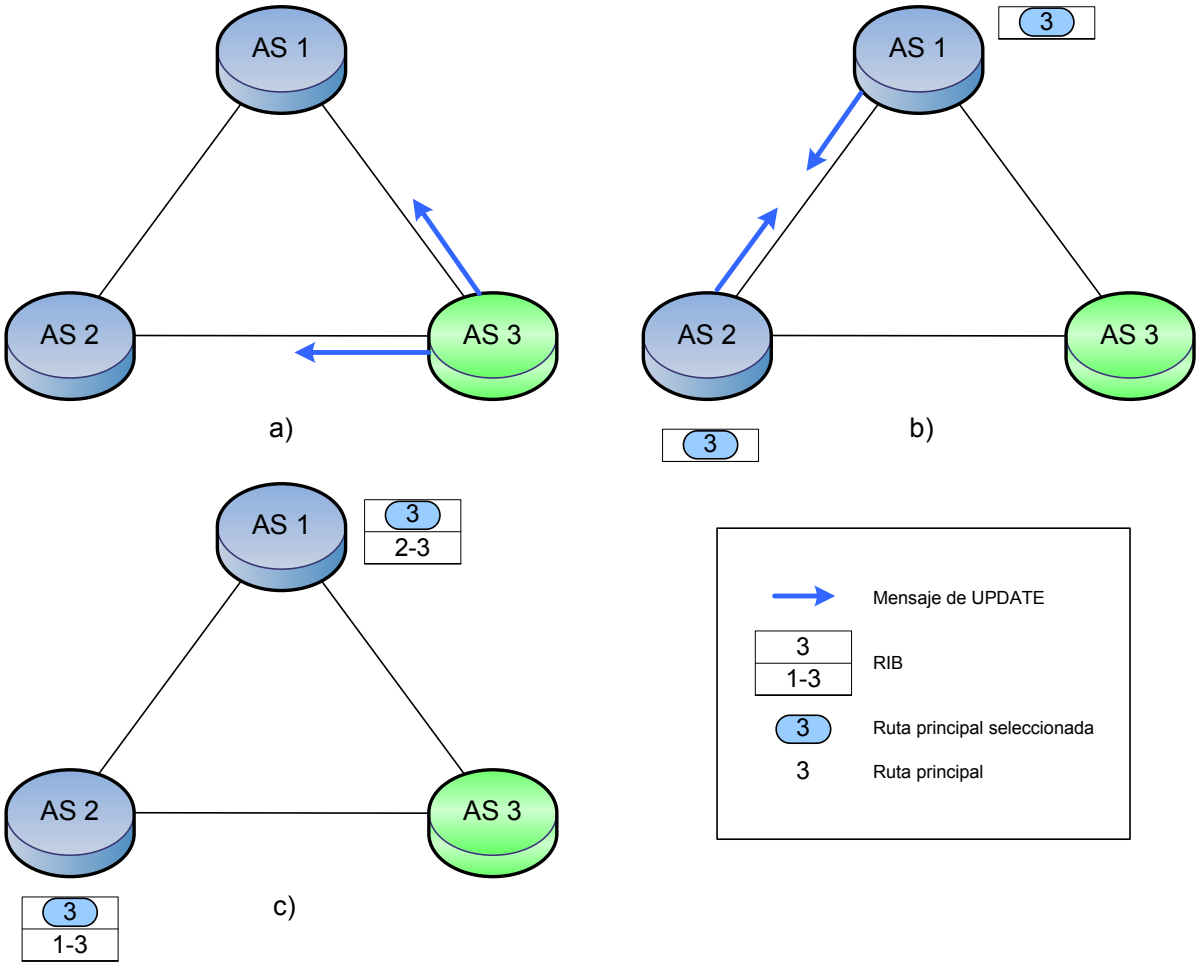


Figura 5.3: Escenario 1: Propagación de rutas con BGP original.

5.1.2. Resultados

A continuación se presentarán los resultados obtenidos tras la simulación del escenario 1 utilizando la nueva implementación de BGP sobre quagga. Para comprender mejor los valores que aparecen en las tablas de prefijos de cada uno de los *peers* en la figura 5.5 se muestra la topología de la red acompañada de las direcciones de cada sub-red e interfaces de las mismas.

Para verificar el correcto funcionamiento del mecanismo implementado se ha realizado la captura de paquetes sobre la red simulada utilizando un *software* apropiado, en este caso se ha utilizado *wireshark*[14]. Para realizar la selección de paquetes capturados se ha capturado cualquier mensaje que circulara por la red y posteriormente en el filtro de *wireshark* se ha fijado el

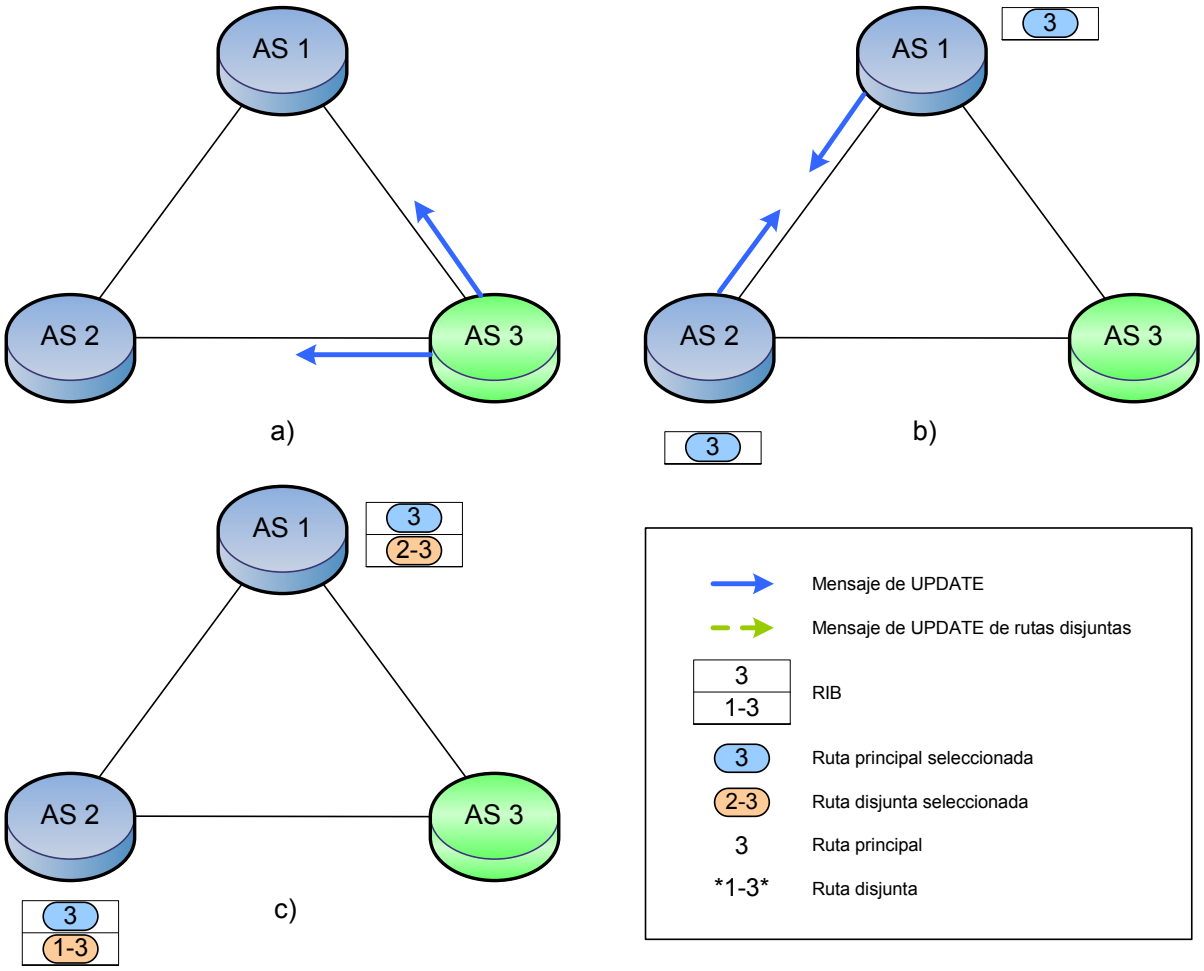


Figura 5.4: Escenario 1: Propagación de rutas con BGP modificado.

siguiente valor:

```
filter: bgp.type == 2
```

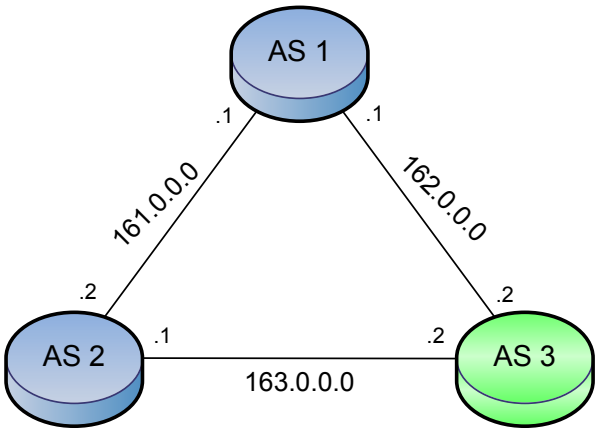


Figura 5.5: Escenario 1: Topología de red con direcciones.

Este valor en el filtro hace que sólo se muestren los paquetes del protocolo BGP y, de forma más específica, aquellos que tienen como valor del campo *tipo* 2, es decir, sean mensajes *update* de BGP. Esta captura de paquetes de realizará de la misma forma en todos los escenarios y los mensajes capturados en cada una de las simulaciones se encuentra en el directorio adjunto en el disco de este proyecto cuya ruta es: `/simulaciones/capturas/`. En esta carpeta se encuentran las capturas de todos los escenarios y modos de funcionamiento simulados en el proyecto.

El siguiente paso para verificar el funcionamiento del mecanismo de selección de rutas implementado será la visualización de las RIBs de cada uno de los *peers* de la red. En el apartado 6 del apéndice C (*Monitorización de BGP*) se muestra la forma de obtener estas tablas de prefijos. En las tablas se puede ver una columna que indica "Tipo de ruta" en la que P indica una ruta principal y S una secundaria, los únicos campos que es necesario aclarar son las dos últimas columnas: *Princ.* marca con * la mejor ruta principal seleccionada y de igual forma, *Secund.* marca la mejor ruta secundaria disjunta seleccionada. De la misma forma que se hizo con la captura de paquetes de *Wireshark* las RIBs obtenidas utilizando el virtualizador VNUML se encuentran almacenadas en el directorio `/simulaciones/RIBs/` del cd adjunto.

Destino	Tipo de ruta	Siguiente salto	Ruta	Princ.	Secund.
103.0.0.0	P	161.0.0.2	2 - 3		*
	P	162.0.0.2	3	*	

Tabla 5.1: Escenario 1: *Adj-RIB-In* del AS1.

Destino	Tipo de ruta	Siguiente salto	Ruta	Princ.	Secund.
103.0.0.0	P	161.0.0.1	1 - 3		*
	P	163.0.0.2	3	*	

Tabla 5.2: Escenario 1: *Adj-RIB-In* del AS2.

En las tablas 5.1 y 5.2 se puede comprobar como las entradas de cada una de ellas se corresponden con los valores obtenidos en la simulación teórica de este escenario que se ha analizado en la figura 5.4. No se muestra la tabla de prefijos del AS3 puesto que este *peer* conoce desde el comienzo la ruta para alcanzar el destino que esta en su propio Sistema Autónomo.

Para realizar la simulación de cada uno de los escenarios propuestos en este proyecto se ha ejecutado dicha simulación en 20 ocasiones independientes y posteriormente se ha realizado un estudio estadístico utilizando intervalos de confianza para verificar la veracidad de los resultados obtenidos y poder trabajar con ellos en consecuencia.

Los tiempos obtenidos para cada una de las simulaciones se pueden consultar en el fichero adjunto *TiemposEscenario1.pdf* que se encuentra en el directorio `/simulaciones/tiempos/` del cd adjunto.

Protocolo BGP utilizado	Media	Desviación típica
Normal	27,1484	1,56
Modificado	28,9118	1,76722

Tabla 5.3: Escenario 1: Tiempo medio y desviación típica.

En la tabla 5.3 se pueden observar las medias y desviaciones típicas obtenidas para el escenario 1 en la utilización del protocolo BGP normal y modificado. Antes de analizar los resultados obtenidos se realizará el intervalo de confianza de los mismos para verificar que los resultados son “fiables”.

Si se calculan estos intervalos para ambos casos con un nivel de confianza del 95 %, teniendo en cuenta las 20 simulaciones realizadas y sus respectivas desviaciones típicas se obtienen los

intervalos presentados en la tabla 5.4.

BGP utilizado	Incremento obtenido	Intervalo obtenido
Normal	0,6832 = 2,52 %	27,1484 ± 0,6832
Modificado	0,7745 = 2,68 %	28,9118 ± 0,7745

Tabla 5.4: Escenario 1: Intervalos de confianza obtenidos.

Como se observa en la tabla 5.4 tanto para BGP normal como para el modificado, tras realizar las simulaciones se resuelve que, con un 95 % de confianza, tendrán sus valores comprendidos entorno al 2,5 % por encima o debajo del valor de la media. Este resultado nos indica que los valores de tiempo de ejecución obtenidos tras las simulaciones son correctos con un 95 % de confianza y pueden ser utilizados como tales.

Por último y tras verificar la validez de los tiempos obtenidos en las simulaciones, en la tabla 5.5 se puede observar como los tiempos medios de ejecución en ambos casos son muy parecidos. Esto se debe a lo que se acaba de explicar en el apartado anterior, en este caso no es necesario el intercambio de información adicional lo que provoca que el tiempo de ejecución del mecanismo de propagación de rutas sea prácticamente el mismo. Como se observa el protocolo BGP normal tarda 27,1483 segundos en propagar las rutas mientras que la modificación que se ha implementado tarda 28,9117 segundos. Se ha producido un incremento de 1,7634 segundos que supone un 6,5 % del tiempo utilizado por el protocolo BGP original.

	Tiempo de ejecución (seg)
BGP original	27,1483
BGP modificado	28,9117

Tabla 5.5: Escenario 1: Tiempos de ejecución.

Como se ha podido observar en la explicación anterior sobre el funcionamiento de este escenario, los mensajes intercambiados utilizando ambas versiones de BGP son los mismos, es decir, en BGP modificado no se envía ningún mensaje *update* de ruta secundaria. Únicamente el tiempo sufre un ligero incremento debido al tiempo que cada *peer* tarda en procesar las rutas recibidas, comprobar que no sean disjuntas con la principal y posteriormente fijarlas como rutas secundarias en la *Loc-RIB*. Este primer escenario es muy simple por lo que la modificación introducida en BGP no supone una gran ínversión de tiempo para los *peers*.

5.2. ESCENARIO 2

En este escenario se incrementa levemente la complejidad de la red introduciendo un enlace intermedio (entre AS2 y AS3) y un AS más que en el escenario anterior. La red de este escenario aparece en la figura 5.6. De la misma forma que sucederá en todos los escenario, el nodo pintado de verde es hacia el que vaya destinada la información de alcanzabilidad intercambiada y el resto son de color azul.

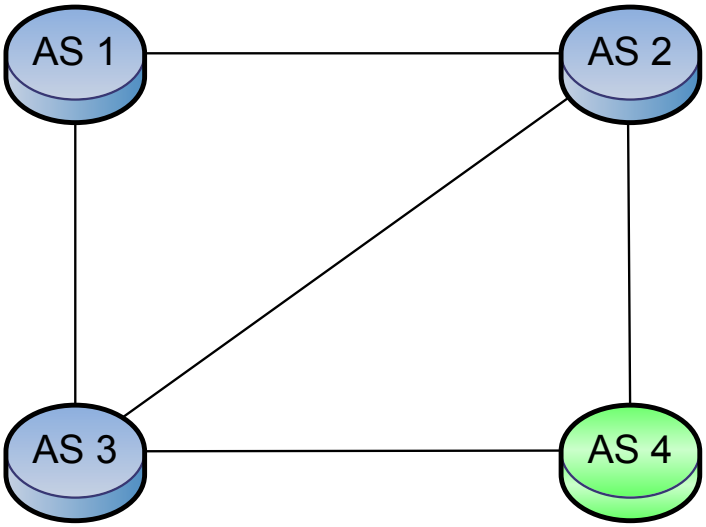


Figura 5.6: Escenario 2: Topología de la red.

De la misma manera que se hizo en el escenario anterior, se realizará en primer lugar el análisis del comportamiento del escenario 2 utilizando el protocolo BGP original y más tarde ese mismo análisis utilizando el protocolo BGP modificado en este proyecto.

5.2.1. Funcionamiento

En la figura 5.7 se observa la propagación de rutas con destino AS4 que se acaba de mencionar en la introducción anterior.

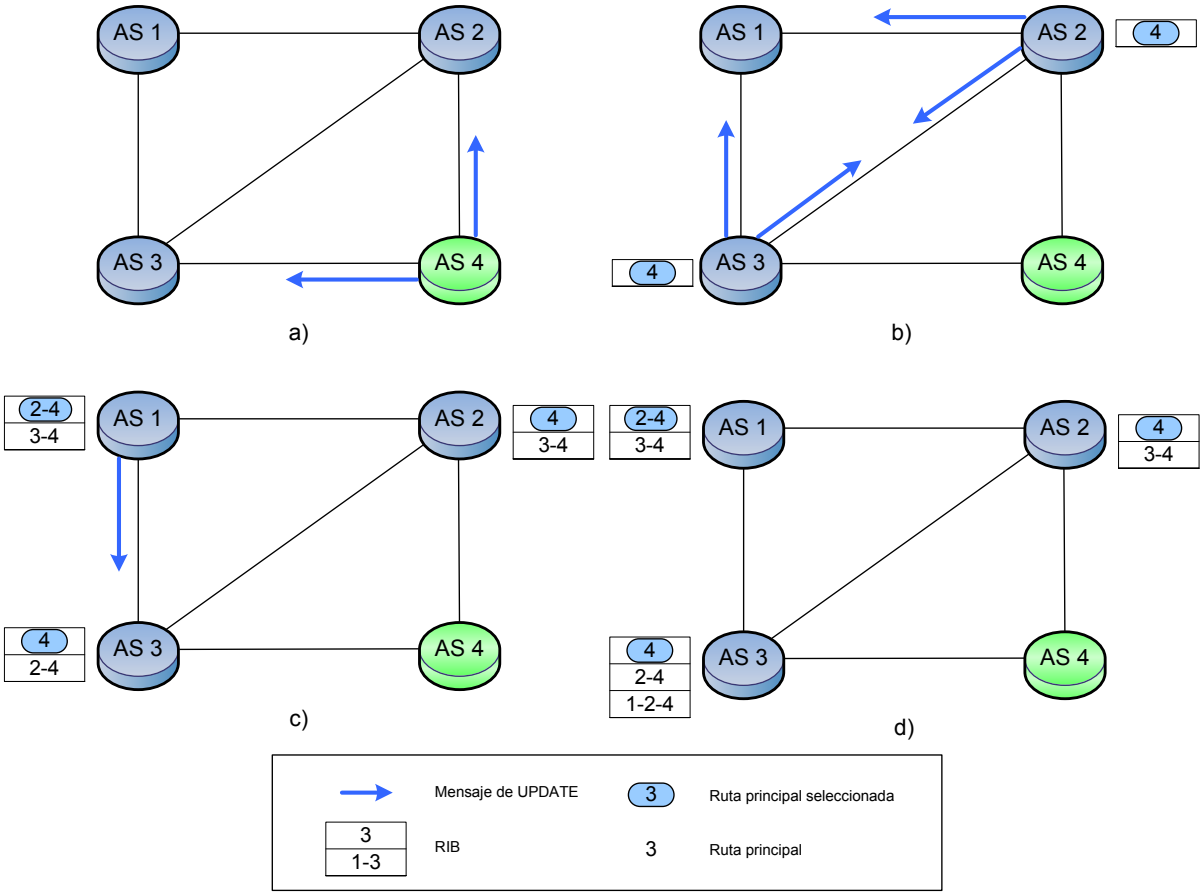


Figura 5.7: Escenario 2: Propagación de rutas con BGP original.

En este caso se observa, en los diferentes apartados de la figura 5.7, como AS4 inicia el envío de mensajes *update* (ap. a), posteriormente AS2 y AS3 propagan sus respectivas rutas aprendidas(ap. b) y por último AS1 notifica su ruta aprendida, a través de AS2, a AS3(ap. c). En el apartado d se puede observar un estado final en el que todos los *peers* de la red han aprendido una ruta para alcanzar el destino en AS4.

En la figura 5.8 se puede observar el intercambio de mensajes producido y el estado de las RIBs de los diferentes *peers* de la red utilizando el protocolo BGP modificado en este proyecto.

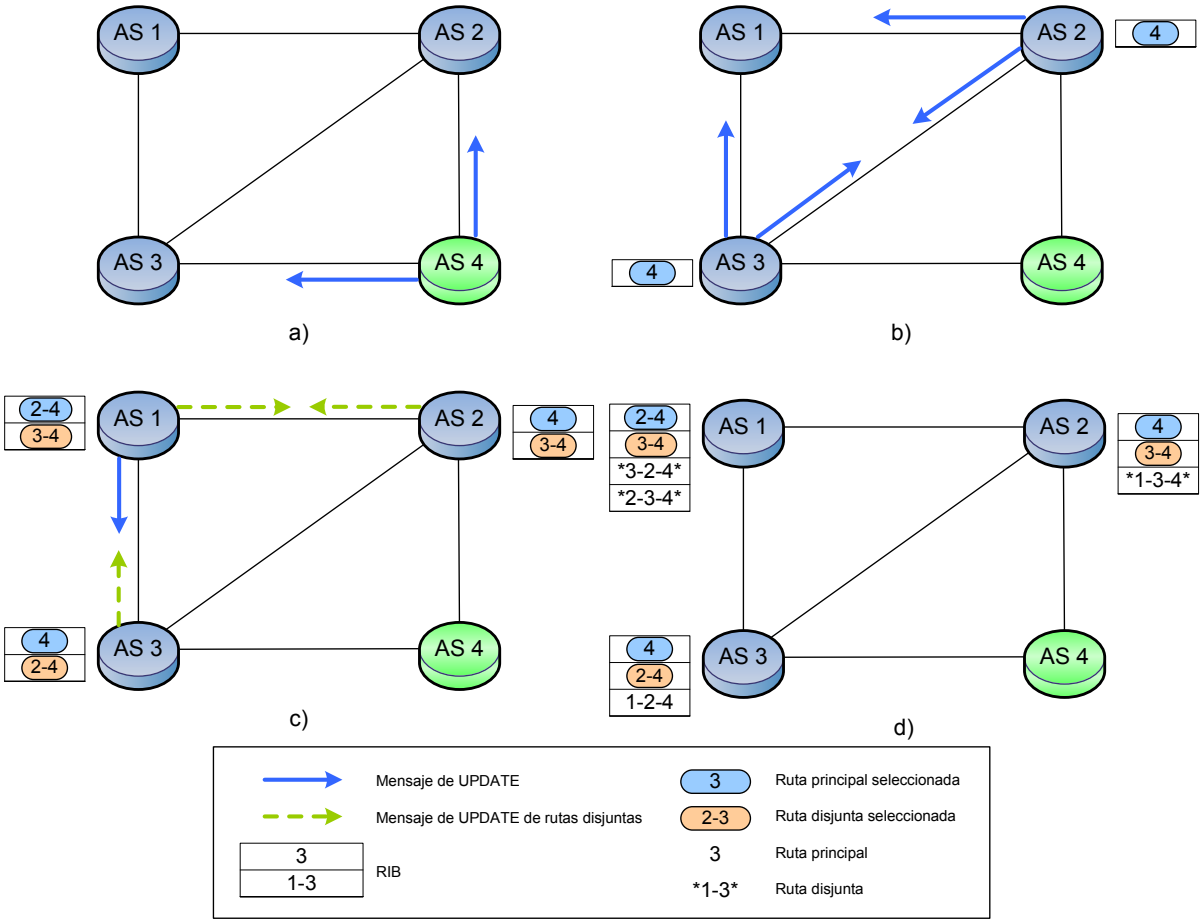


Figura 5.8: Escenario 2: Propagación de rutas con BGP modificado.

Se puede observar como el intercambio de mensajes en los apartados a y b de la figura 5.8 son idénticos pero en el apartado c se produce el intercambio de 3 mensajes *update* de ruta secundaria. Esto se debe a que AS1, AS2 y AS3 han encontrado rutas disjuntas a la principal en sus respectivas *Adj-RIB-In* y, a consecuencia de esto, han introducido las nuevas rutas secundarias en sus *Adj-RIB-Out*. Al introducir estas nuevas rutas en la *Adj-RIB-Out* cada uno de ellos envía un mensaje *update* de ruta secundaria (ap. c) a sus vecinos correspondientes. En el apartado d se muestra como todos los *peers* de la red poseen dos rutas para alcanzar el destino en AS4[1].

[1]Ruta principal marcada en azul y la ruta secundaria en salmón.

5.2.2. Resultados

A continuación se presentarán los resultados obtenidos tras la simulación de este segundo escenario que consta de 4 *peers* por lo que se incrementa un poco la complejidad de la red y por tanto de la simulación. De la misma forma que sucedía en el escenario anterior, para comprender mejor los valores que aparecen en las tablas de prefijos de cada uno de los *peers* en la figura 5.9 se muestra la topología de la red acompañada de las direcciones de cada sub-red e interfaces de las mismas.

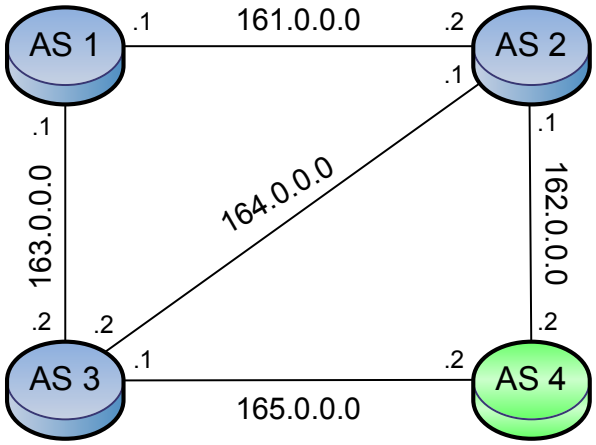


Figura 5.9: Escenario 2: Topología de red con direcciones.

En todos los escenarios que se han simulado en este proyecto se ha utilizado *Wireshark* para realizar la captura de paquetes BGP así como las tablas de prefijos de la implementación de VNUML para obtener la información de las RIBs de cada uno de los *peers* de la red. Como ya se ha comentado anteriormente, tanto las capturas como las tablas de prefijos se encuentra almacenadas en el disco adjunto para su análisis si se considerara oportuno.

Se utilizará de aquí en adelante el mismo formato de tablas que se utilizó en el escenario anterior para visualizar el estado final de las listas de prefijos de cada uno de los *peers*. En las siguientes tablas se pueden observar los valores obtenidos al final de la simulación.

Destino	Tipo de ruta	Siguiente salto	Ruta	Princ.	Secund.
103.0.0.0	P	161.0.0.2	2 - 4	*	
	P	163.0.0.2	3 - 4		*
	S	161.0.0.2	2 - 3 - 4		
	S	163.0.0.2	3 - 2 - 4		

Tabla 5.6: Escenario 2: *Adj-RIB-In* del AS1.

Destino	Tipo de ruta	Siguiente salto	Ruta	Princ.	Secund.
103.0.0.0	P	161.0.0.1	4	*	
	P	164.0.0.2	3 - 4		*
	S	162.0.0.2	1 - 3 - 4		

Tabla 5.7: Escenario 2: *Adj-RIB-In* del AS2.

Destino	Tipo de ruta	Siguiente salto	Ruta	Princ.	Secund.
103.0.0.0	P	165.0.0.2	4	*	
	P	164.0.0.1	2 - 4		*
	P	163.0.0.1	1 - 2 - 4		

Tabla 5.8: Escenario 2: *Adj-RIB-In* del AS3.

En las tablas 5.6, 5.7 y 5.8 se puede comprobar como las entradas de cada una de ellas se corresponden con los valores obtenidos en la simulación teórica de este escenario que se ha analizado en la figura 5.8. De la misma forma que se hará en el resto de escenarios, no se mostrará la tabla de prefijos del *peer* del Sistema Autónomo en el que se encuentra la red destino, en este caso 103.0.0.0, en concreto en este escenario se ha obviado la RIB del AS4.

Para realizar la simulación de este escenario de la misma manera que el anterior, se han ejecutado 20 simulaciones independientes y posteriormente se ha realizado el estudio estadístico utilizando intervalos de confianza para verificar la veracidad de los resultados obtenidos y poder trabajar con ellos en consecuencia.

Los tiempos obtenidos para cada una de las simulaciones se pueden consultar en el fichero adjunto *TiemposEscenario2.pdf* que se encuentra en el directorio `/simulaciones/tiempos/` del cd adjunto.

Protocolo BGP utilizado	Media	Desviación típica
Normal	30,8140	1,68
Modificado	59,7725	1,42831

Tabla 5.9: Escenario 2: Tiempo medio y desviación típica.

En la tabla 5.9 se pueden observar las medias y desviaciones típicas obtenidas para el escenario 2 en la utilización del protocolo BGP normal y modificado. Antes de analizar los resultados obtenidos se realizará el intervalo de confianza de los mismos para verificar la fiabilidad de los resultados.

Si se calculan estos intervalos de confianza de estas 20 simulaciones para BGP original y modificado con un nivel de confianza del 95 %, se obtienen los intervalos presentados en la tabla 5.10.

BGP utilizado	Incremento obtenido	Intervalo obtenido
Normal	0,7352 = 2,38593 %	30,814 ± 0,7352
Modificado	0,626 = 1,0473 %	59,7725 ± 0,626

Tabla 5.10: Escenario 2: Intervalos de confianza obtenidos.

Como se observa en la tabla 5.10 tanto para BGP normal como para el modificado, tras realizar las simulaciones se resuelve que, con un 95 % de confianza, sus valores estarán comprendidos entorno a la media con un error del 2,38 % y 1 % del valor de la media respectivamente. Este resultado nos indica que los valores de tiempo de ejecución obtenidos para cada caso tras realizar las simulaciones son correctos con un 95 % de confianza y pueden ser utilizados como tales.

En la tabla 5.11 se puede observar como el tiempo de ejecución se ha visto incrementado prácticamente unos 30 segundos del protocolo BGP original al modificado.

	Tiempo de ejecución (seg)
BGP original	30,8140
BGP modificado	59,7725

Tabla 5.11: Escenario 2: Tiempos de ejecución.

La explicación por la que se produce un incremento de 30 segundos al realizar las modificaciones comentadas para la nueva implementación de BGP es muy sencilla, BGP dispone de 5 temporizadores que se encargan de fijar la frecuencia de ejecución de una serie de funciones del protocolo. En concreto y para este caso, se debe centrar la explicación sobre el temporizador denominado *MinRouteAdvertisementInterval*. Este temporizador se encarga de fijar el tiempo mínimo que debe transcurrir entre dos avisos consecutivos de actualización de rutas para un mismo destino y un mismo *peer* vecino. De manera que, si se vuelve a observar la representación gráfica del intercambio de mensajes para la propagación de rutas del protocolo BGP modificado (véase la figura 5.8), se observa como esta situación aparece por partida doble:

- En el paso b de la figura 5.8 el AS2 envía su ruta principal a AS1 y posteriormente, en el paso c, se observa como AS2 envía su ruta secundaria también a AS1. Para realizar este envío de rutas consecutivas del mismo destino al mismo AS el protocolo BGP define que se tienen que esperar los 30 segundos del temporizador mencionado anteriormente.

- La misma situación tiene lugar cuando AS3 envía su ruta principal a AS1 (paso b) y posteriormente, paso c, envía su ruta secundaria a AS1. También deberá esperar esos 30 segundos.

Ante esta situación, se puede observar como el tiempo de ejecución del protocolo BGP modificado será similar al de BGP normal pero con el incremento de esos 30 segundos del temporizador. Una posible solución a este problema consistiría en resetear el valor del contador al enviar dos rutas consecutivas de distinto tipo, es decir, si envía una ruta principal un AS no debería tener que esperar nada para enviar la ruta secundaria, en este caso se enviaría la ruta inmediatamente y se fijaría de nuevo el temporizador a su valor por defecto de 30 segundos. Esta solución se propondrá en los trabajos futuros del proyecto y existe otra posible alternativa para reducir este retardo indeseado. Simplemente habría que reducir el valor de este temporizador (por ejemplo

a 10 segundos), aunque la reducción de ese tiempo no es trivial, habría que estudiar con mayor detalle las consecuencias que conllevaría esta propuesta.

5.3. ESCENARIO 3

En este escenario se incrementa bastante la complejidad de la red a simular. Se introducen en el mismo 7 Sistemas Autónomos y diferentes enlaces que convierten el grafo de la red en 2-conectado. Como se puede ver en la figura 5.10 los mensajes que se intercambian en este caso con la información de alcanzabilidad tienen como destino el AS4, que aparece marcado en verde. La importancia de este apartado reside en el número de rutas que se intercambian y la comprobación de disjunción que se hará en alguno de los casos. Se verá como la modificación propuesta, efectivamente, no propaga rutas secundarias que no sean disjuntas a la ruta principal fijada en la *Loc-RIB*.

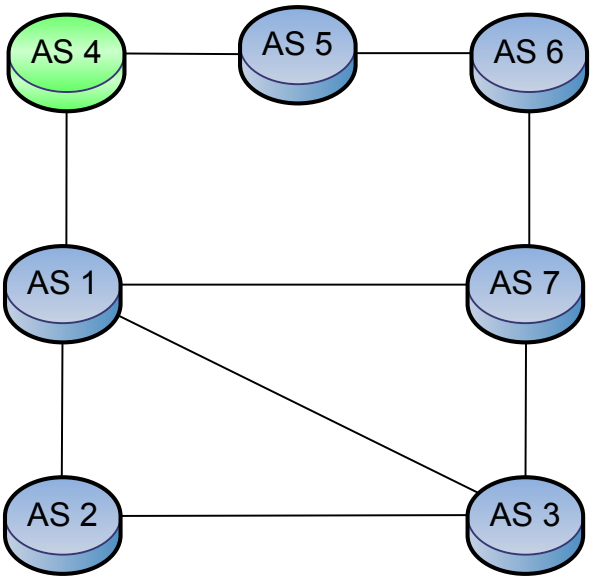


Figura 5.10: Escenario 3: Topología de la red.

5.3.1. Funcionamiento

Como se viene realizando en los apartados anteriores, se llevará a cabo un análisis previo del funcionamiento de este escenario mediante la utilización del protocolo BGP original. La propagación de rutas con destino AS4 que se produciría para esta primera versión de BGP sería la que aparece representada en la figura 5.11.

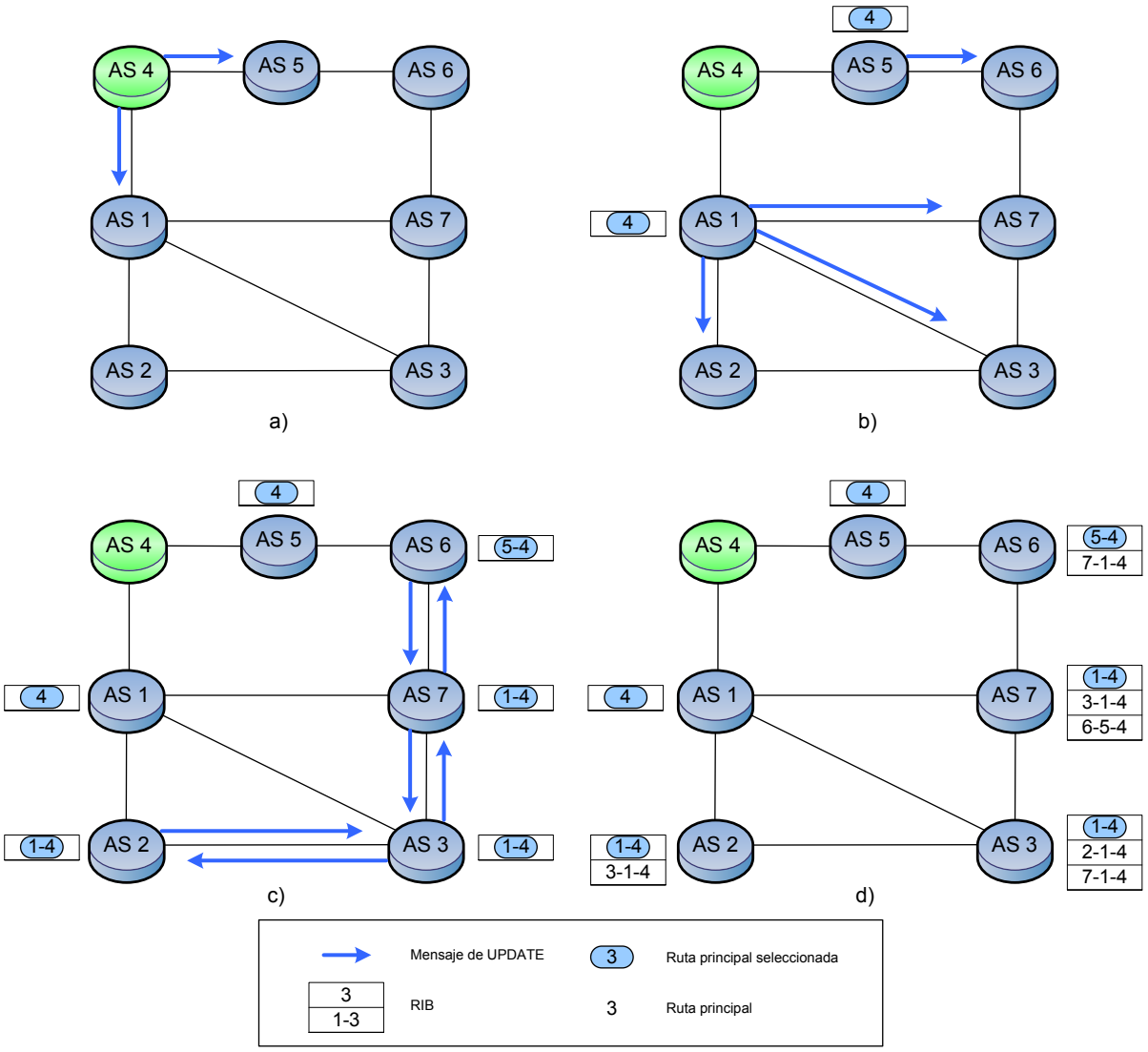
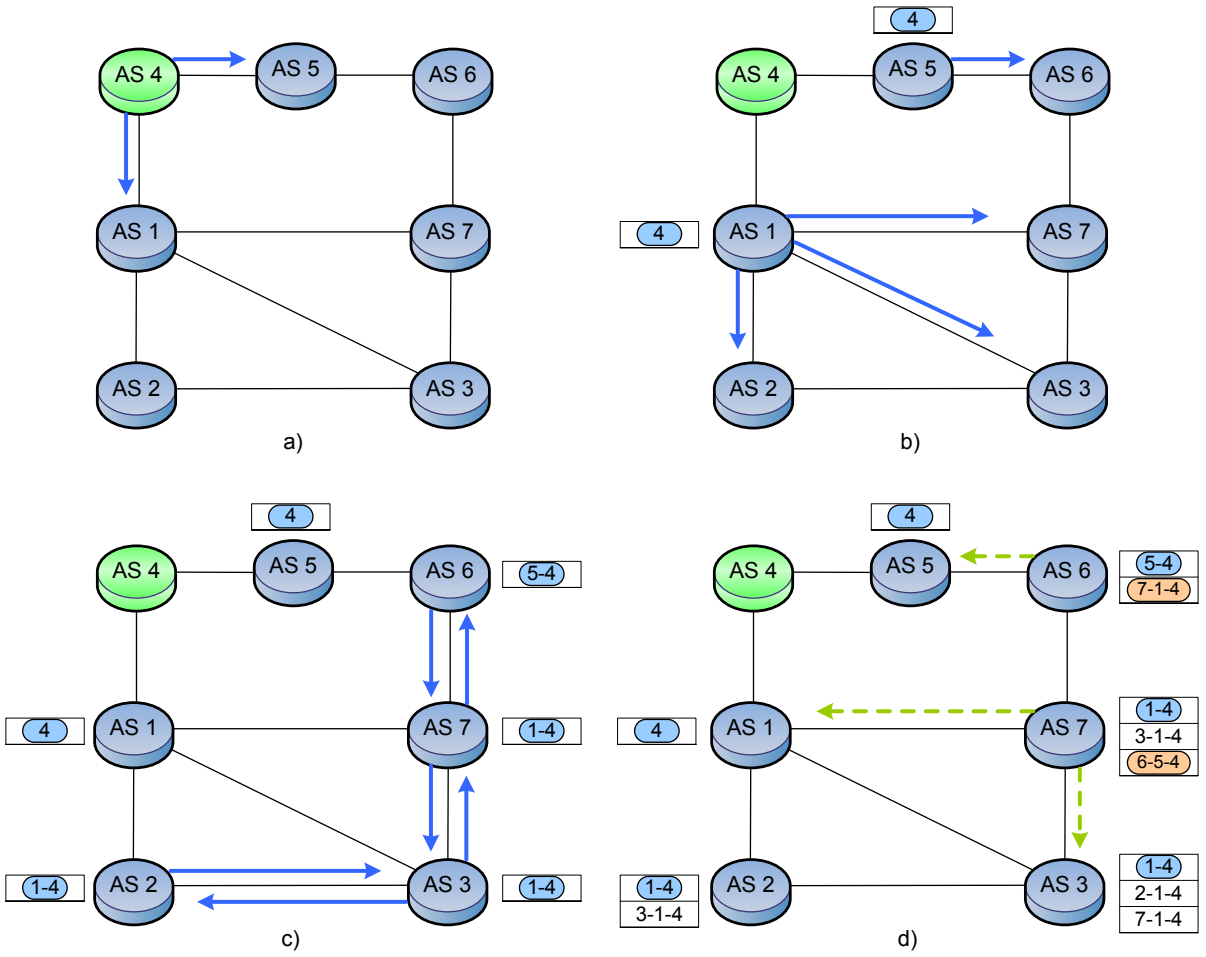


Figura 5.11: Escenario 3: Propagación de rutas con BGP original.

AS4 inicia el mecanismo de propagación enviando su mensaje *update* a sus vecinos AS5 y AS1 (ap. a). A continuación AS5 y AS1 envía su respectivo mensaje *udpate* a sus propios vecinos (ap. b) y así sucesivamente hasta alcanzar la estabilidad de las RIBs de la red (ap. d) donde se puede observar que los 7 *peers* conocen una ruta para alcanzar un destino perteneciente al AS4.

En la figura 5.12 se puede observar el intercambio de mensajes producido y el estado de las RIBs de los diferentes *peers* de la red utilizando el protocolo BGP modificado en este proyecto.



En la secuencia representada en la figura 5.12 se puede observar algo que se viene apreciando en todos los ejemplos realizados, los primeros intercambios de mensajes *update* (ap. a, b y c) son idénticos a los que tenían lugar utilizando el protocolo BGP original. Sin embargo, en este caso se observa en el apartado d como los primeros ASes en obtener una ruta secundaria disjunta a la principal son AS6 y AS7. Serán estos ASes los primeros en añadir un anuncio de ruta secundaria a sus respectivas *Adj-RIB-Out*. En el momento que introducen dicha ruta en esta RIB se envían a todos sus vecinos, excepto al que le envió la ruta que quiere notificar, un mensaje *update* de ruta secundaria seleccionada.

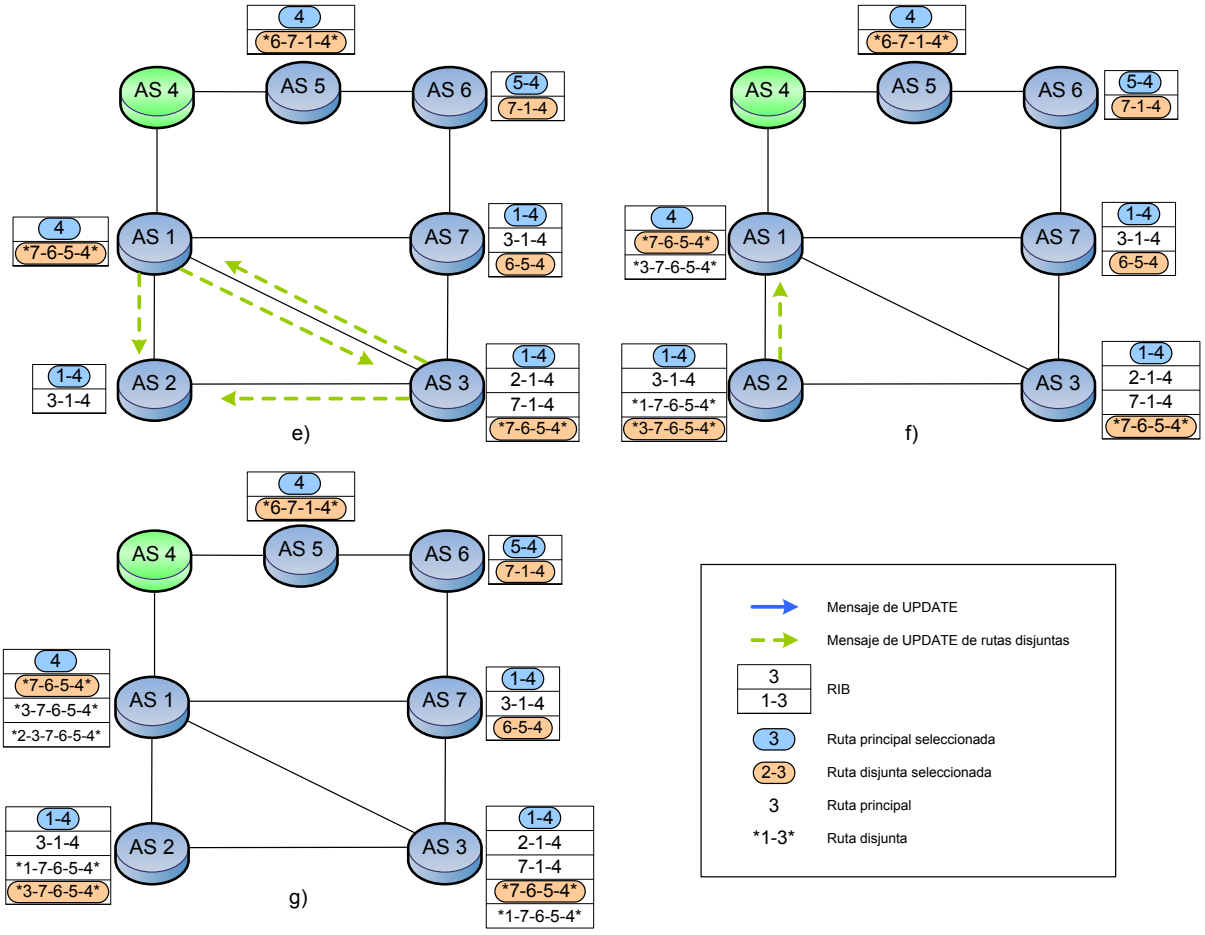


Figura 5.12: Escenario 3: Propagación de rutas con BGP modificado.

Para comprobar el correcto funcionamiento de la implementación que aquí se ha llevado a cabo, se puede observar en el apartado d como AS3 posee tres rutas en su RIB que son: 1-4, 7-1-4 y 2-1-4. Como la ruta principal seleccionada es 1-4, ningunas de las 2 rutas restantes son disjuntas a ella porque tienen el AS1 como AS común en sus rutas, es decir, ninguna de ellas podrá ser seleccionada como ruta secundaria para AS3. En la figura que se ha mencionado se puede ver como AS3 no seleccionada a ninguna de las dos y por lo tanto presenta un funcionamiento de acuerdo a las especificaciones del modelo propuesto en el capítulo 7 de [1].

En el apartado e de la figura 5.12 se observa como se siguen asignando rutas secundarias y enviando sus respectivos mensajes *update*. En este apartado cabe destacar la asignación de ruta

secundaria que realiza AS3, se observa como recibe una nueva ruta secundaria (*7-6-5-4*) de su vecino AS7. Esta ruta si es disjunta con respecto a su principal (1-4) por lo que puede fijarla como ruta secundaria y enviar el mensaje *update* de la misma a sus vecinos AS1 y AS2 (ap. e). Posteriormente AS1 no utiliza esa ruta recibida pero AS2 sí, éste envía su respectivo mensaje *update* de ruta secundaria a AS1 (ap. f) y tras ese envío se alcanza la estabilidad de las listas de prefijos de la red (ap. g) teniendo todos los *peers* de la misma una ruta principal y secundaria disjunta para alcanzar un destino de AS4.

5.3.2. Resultados

A continuación se presentarán los resultados obtenidos tras la simulación del tercer escenario que consta de 7 *peers* por lo que se incrementa considerablemente la complejidad de la red y por tanto de la simulación. De la misma forma que sucedía en los escenarios anteriores, para comprender mejor los valores que aparecen en las tablas de prefijos de cada uno de los *peers* en la figura 5.13 se muestra la topología de la red acompañada de las direcciones de cada sub-red e interfaces de las mismas.

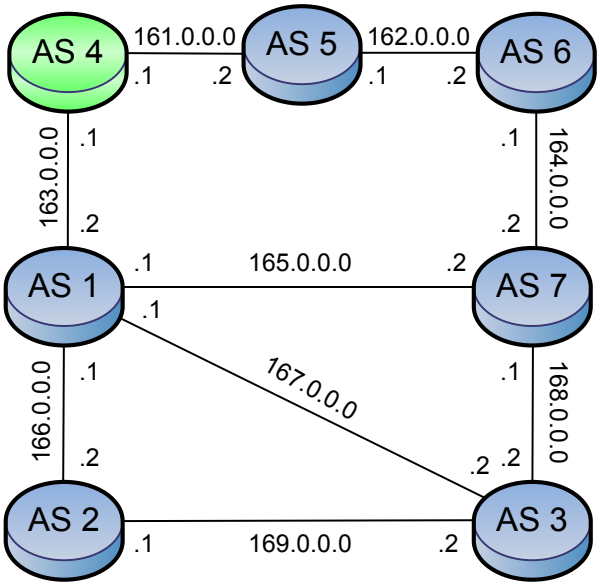


Figura 5.13: Escenario 3: Topología de red con direcciones.

En todos los escenarios que se han simulado en este proyecto se ha utilizado *Wireshark* para

realizar la captura de paquetes BGP así como las tablas de prefijos de la implementación de VNUML para obtener la información de las RIBs de cada uno de los *peers* de la red. Como ya se ha comentado anteriormente, tanto las capturas como las tablas de prefijos se encuentra almacenadas en el disco adjunto para su análisis si se considerara oportuno.

Se utilizará de aquí en adelante el mismo formato de tablas que se utilizó en el escenario anterior para visualizar el estado final de las listas de prefijos de cada uno de los *peers*. En las siguientes tablas se pueden observar los valores obtenidos al final de la simulación.

Destino	Tipo de ruta	Siguiente salto	Ruta	Princ.	Secund.
103.0.0.0	P	163.0.0.1	4	*	
	S	165.0.0.2	7 - 6 - 5 - 4		*
	S	167.0.0.2	3 - 7 - 6 - 5 - 4		
	S	166.0.0.2	2 - 3 - 7 - 6 - 5 - 4		

Tabla 5.12: Escenario 3: *Adj-RIB-In* del AS1.

Destino	Tipo de ruta	Siguiente salto	Ruta	Princ.	Secund.
103.0.0.0	P	166.0.0.1	1 - 4	*	
	P	169.0.0.2	3 - 1 - 4		
	S	169.0.0.2	3 - 7 - 6 - 5 - 4		*
	S	166.0.0.1	1 - 7 - 6 - 5 - 4		

Tabla 5.13: Escenario 3: *Adj-RIB-In* del AS2.

Destino	Tipo de ruta	Siguiente salto	Ruta	Princ.	Secund.
103.0.0.0	P	167.0.0.1	1 - 4	*	
	P	169.0.0.1	2 - 1 - 4		
	P	168.0.0.1	7 - 1 - 4		
	S	168.0.0.1	7 - 6 - 5 - 4		*
	S	167.0.0.1	1 - 7 - 6 - 5 - 4		

Tabla 5.14: Escenario 3: *Adj-RIB-In* del AS3.

Destino	Tipo de ruta	Siguiente salto	Ruta	Princ.	Secund.
103.0.0.0	P	161.0.0.1	4	*	
	S	162.0.0.2	6 - 7 - 1 - 4		*

Tabla 5.15: Escenario 3: *Adj-RIB-In* del AS5.

Destino	Tipo de ruta	Siguiente salto	Ruta	Princ.	Secund.
103.0.0.0	P	162.0.0.1	5 - 4	*	
	P	164.0.0.2	7 - 1 - 4		*

Tabla 5.16: Escenario 3: *Adj-RIB-In* del AS6.

Destino	Tipo de ruta	Siguiente salto	Ruta	Princ.	Secund.
103.0.0.0	P	165.0.0.1	1 - 4	*	
	P	164.0.0.1	6 - 5 - 4		*
	P	168.0.0.2	3 - 1 - 4		

Tabla 5.17: Escenario 3: *Adj-RIB-In* del AS7.

En las tablas 5.12, 5.13, 5.14, 5.15, 5.16 y 5.17 se puede comprobar como las entradas de cada una de ellas se corresponden con los valores obtenidos en la simulación teórica de este escenario que se ha analizado en la figura 5.12. De la misma forma que se hará en el resto de escenarios, no se ha mostrado la tabla de prefijos del *peer* AS4 por tratarse del Sistema Autónomo en el que se encuentra la red destino.

De la misma forma que los anteriores, se han realizado 20 simulaciones de este escenario y se ha realizado el estudio estadístico de intervalos de confianza para verificar la veracidad de los resultados obtenidos.

Los tiempos obtenidos para cada una de las simulaciones se pueden consultar en el fichero adjunto *TiemposEscenario3.pdf* que se encuentra en el directorio `/simulaciones/tiempos/` del cd adjunto.

Protocolo BGP utilizado	Media	Desviación típica
Normal	60,2936	1,5
Modificado	92,1794	3,28975

Tabla 5.18: Escenario 3: Tiempo medio y desviación típica.

En la tabla 5.18 se pueden observar las medias y desviaciones típicas obtenidas para el escenario 3 en la utilización del protocolo BGP normal y modificado. Antes de analizar los resultados obtenidos se realizará el intervalo de confianza de los mismos para verificar la fiabilidad de los resultados.

Si se calculan los intervalos de confianza de estas 20 simulaciones para cada implementación de BGP con un nivel de confianza del 95 %, se obtienen los intervalos presentados en la tabla 5.19.

BGP utilizado	Incremento obtenido	Intervalo obtenido
Normal	0,659 = 1,09298 %	60,2936 ± 0,659
Modificado	1,4410 = 1,56326 %	92,1794 ± 1,4410

Tabla 5.19: Escenario 3: Intervalos de confianza obtenidos.

Como se observa en la tabla 5.19 tanto para BGP normal como para el modificado, tras realizar las simulaciones se resuelve que, con un 95 % de confianza, sus valores estarán comprendidos entorno a la media con un error del 1 % del valor de la media. Este resultado nos indica que los valores de tiempo de ejecución obtenidos para cada caso tras realizar las simulaciones son muy próximos entre sí y pueden ser utilizados sin problemas.

En la tabla 5.20 se puede observar los tiempos de ejecución de cada implementación de BGP. De nuevo se observa un incremento entorno a los 30 segundos que vuelve a llamar la atención.

	Tiempo de ejecución (seg)
BGP original	60,2936
BGP modificado	92,1794

Tabla 5.20: Escenario 3: Tiempos de ejecución.

De la misma forma que sucedía en el apartado anterior, este incremento de 30 segundos en el tiempo de ejecución puede deberse al temporizador *MinRouteAdvertisementInterval* que utiliza BGP para fijar el tiempo entre actualizaciones de rutas para un mismo destino y un mismo *peer* vecino. Pero para poder afirmar este hecho habrá que comprobarlo a continuación.

Si se recuerda el mecanismo de propagación de rutas de la nueva implementación de BGP para este escenario (véase la figura 5.12) se observa que esta situación de espera únicamente se produce en un momento determinado:

- En el paso c de la figura 5.12 AS7 envía su ruta principal a AS3 y, posteriormente, al encontrar una ruta secundaria, debe enviar de nuevo a AS3 esta ruta secundaria. Para poder realizar este segundo envío deberá esperar esos 30 segundos del temporizador.

En este caso se puede observar como se ha incrementado considerablemente la complejidad de la red, ya que se ha pasado de redes de 3 y 4 *peers* (escenarios 1 y 2) a una red con 7 *peers* y múltiples enlaces entre ellos. Pero a la vista de los resultados obtenidos en la tabla 5.20 se observa que el tiempo de ejecución únicamente se ha visto incrementado en esos 30 segundos del temporizador. De esta manera queda demostrada la ejecución en segundo plano que se ha implementado en este proyecto para que el mecanismo de propagación de rutas secundarias afecte lo mínimo posible al tiempo de ejecución del mecanismo habitual de BGP.

5.4. ESCENARIO 4

En este escenario la complejidad es muy similar al escenario 3. Se tienen 6 Sistemas Autónomos y diferentes enlaces que convierten el grafo de la red en 2-conectado. Como se puede ver en la figura 5.14, la dirección destino se encontrará en el AS1, que aparece marcado en verde. Habrá que prestar especial atención a esta topología porque presenta un funcionamiento diferente a las demás, se podría denominar como topología trampa. Cuando se dice trampa significa que dependiendo del valor de determinados campos utilizados en las reglas de desempate de BGP (*MED*, *local_preference*,...) se podrán obtener todas las rutas principales y secundarias de la red o no. En este apartado se explicará mediante los gráficos pertinentes este posible estado indeseado.

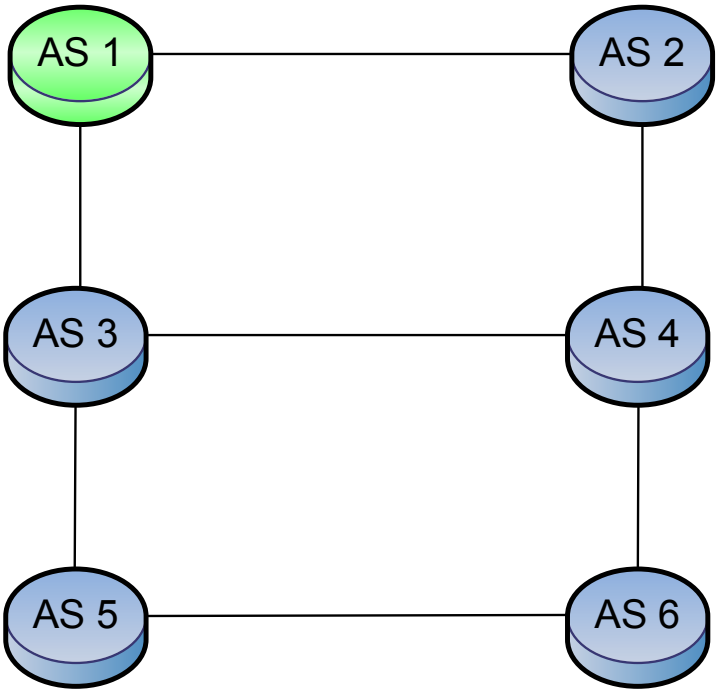
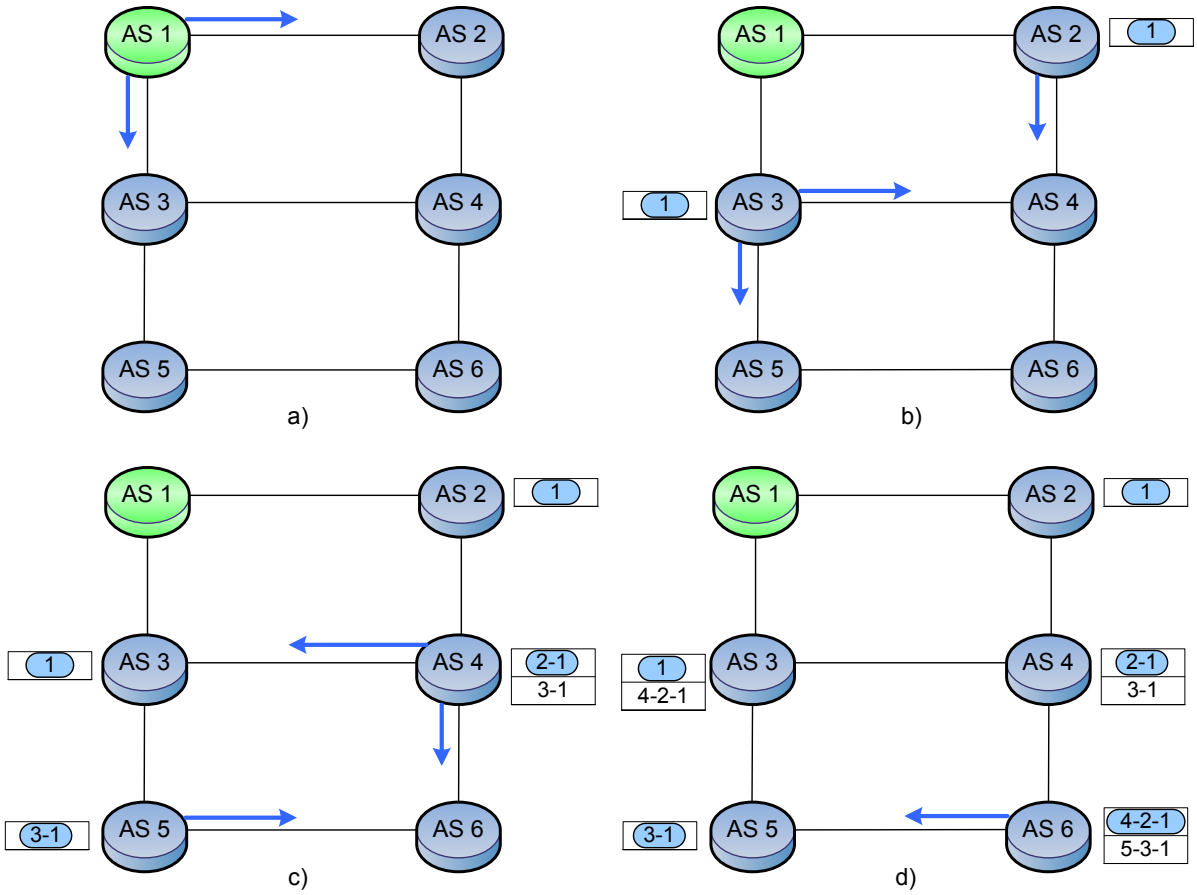


Figura 5.14: Escenario 4: Topología de la red.

En este escenario se realizará la propagación de rutas con destino AS1, marcado en verde en la figura 5.14, y el resto de ASes (en azul) deberán intercambiar su información de alcanzabilidad hasta encontrar la ruta principal y secundaria hacia ese destino.

5.4.1. Funcionamiento

En primer lugar se analiza el comportamiento de la topología trampa que aquí se ha propuesto utilizando el protocolo BGP original. Para esta primera simulación el intercambio de mensajes y estado de las RIBs de cada *peer* de la red está representado en la figura 5.15.



En la figura 5.15 se puede observa como esta topología no presenta ningún problema en especial. Simplemente se observa que AS1 es el Sistema Autónomo que inicia el intercambio de mensajes *update*, enviando a sus vecinos AS2 y AS3 la ruta hacia su destino (ap. a). En el apartado b se puede ver como AS2 y AS3 notifican a sus vecinos (AS 4 para AS2, y AS4 y AS5 para AS3) la agregación de una nueva ruta en sus respectivas *Loc-RIB* enviandoles mensajes *update* con esta información. Este mismo proceso se repite en los apartados c y d como se puede observar en la misma figura.

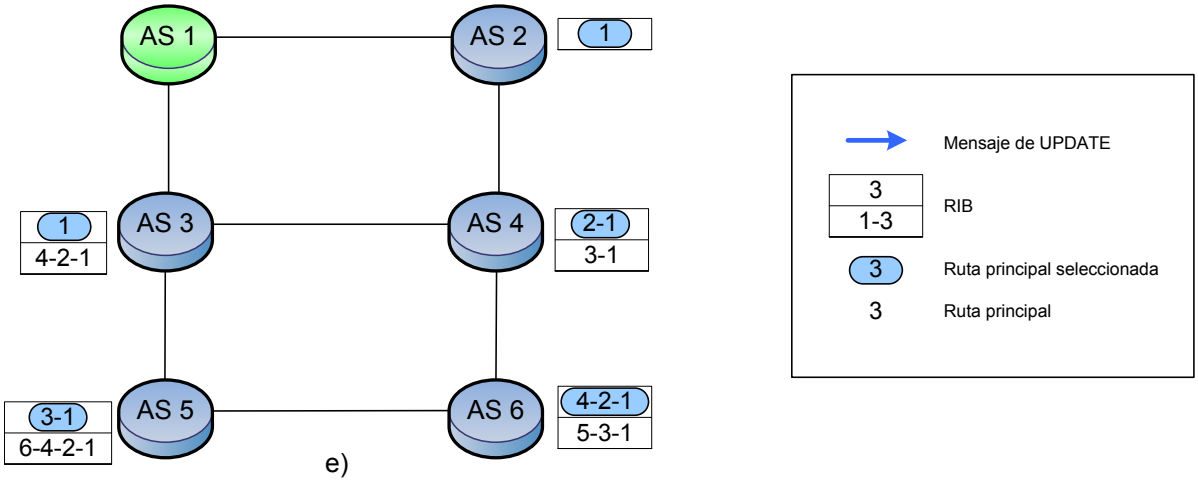
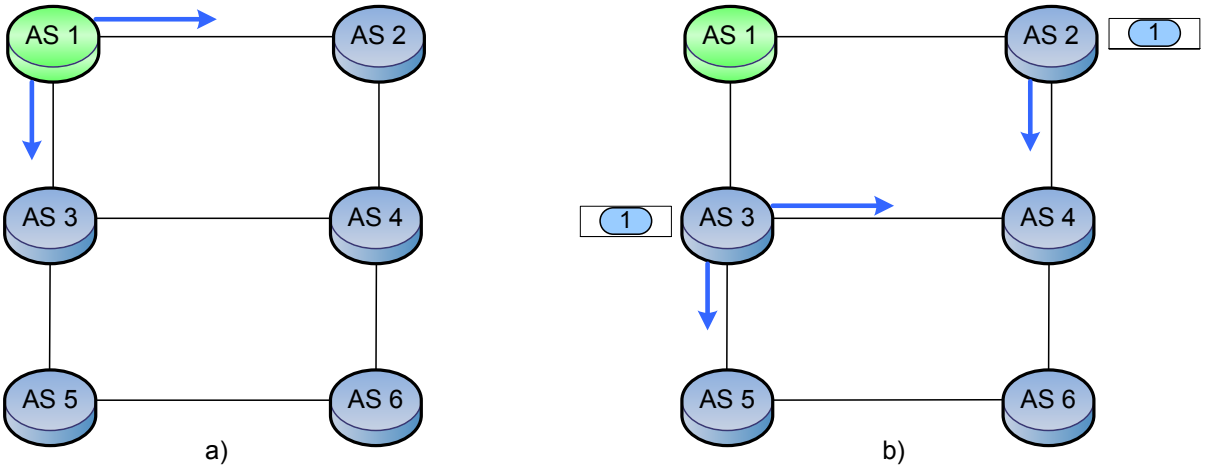


Figura 5.15: Escenario 4: Propagación de rutas con BGP original.

Finalmente, en el apartado e de la figura 5.15 se puede observar un estado final de la simulación totalmente normal en el que todos los *peers* de la red han encontrado una ruta para alcanzar el destino en el que se encuentra en AS1. Esto nos indica que esta topología, para el protocolo BGP original, no presenta ningún problema. Sin embargo, en la simulación de este mismo escenario con el protocolo BGP modificado se observan algunos problemas.

El siguiente paso será realizar esta simulación utilizando el protocolo BGP modificado, aquí se mostrará el motivo por el cual la estructura de esta red recibe el nombre de topología trampa.



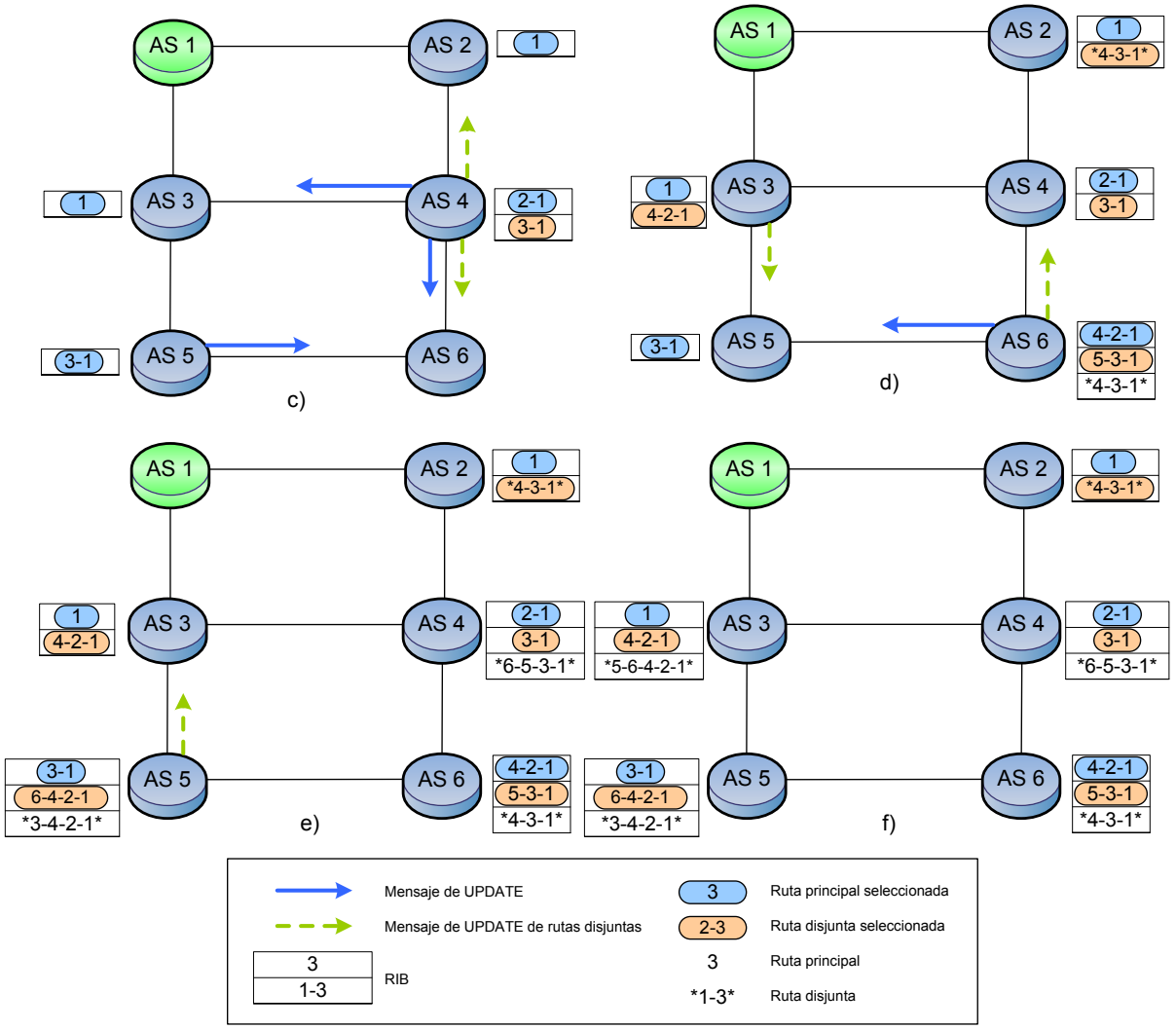


Figura 5.16: Escenario 4: Propagación de rutas con BGP modificado.

En la representación de la figura 5.16 se puede observar un hecho que viene sucediendo en todos los escenarios que han sido simulados. Se puede apreciar como en los 4 primeros apartados de la figura 5.15 y 5.16 el envío de mensajes *update* de ruta principal es exactamente el mismo: AS1 envía su mensaje *update* a sus vecinos AS2 y AS3. Estos a su vez se lo envían a sus respectivos vecinos AS4 y AS5 y estos a su vez envían el suyo a AS6 completando la propagación de rutas con destino en AS1.

Este escenario presenta varios puntos significativos que se analizarán a continuación. En

primer lugar se puede observar en este y los anteriores escenarios la ejecución en paralelo del mecanismo de propagación de rutas secundarias, es decir, se puede ver como en el apartado c de la figura 5.16 el AS4 encuentra una ruta secundaria (3-1) y envía el mensaje *update* de la misma a sus vecinos AS2 y AS6. Lo mismo ocurre en el apartado d cuando AS2 encuentra su ruta secundaria (*4-3-1*), AS3 también encuentra la ruta (4-2-1) como secundaria disjunta y AS6 encuentra sus rutas principal y secundaria simultaneamente (4-2-1) y (5-3-1) respectivamente. Cada uno de ellos enviará su respectivo mensaje *update* de ruta secundaria para notificar esta asignación. En este caso se observa que la ejecución del mecanismo de propagación de rutas secundarias tiene un tiempo de ejecución superior al de ejecución del mecanismo de BGP original, ocupando un paso más en la simulación del mismo.

El segundo punto clave de este escenario hace referencia al estado alcanzado por las tablas de prefijos de cada uno de los *peers* en el apartado f de la figura 5.16 donde se puede observar como todos los *peers* de la red consiguen fijar una ruta principal y secundaria disjunta para alcanzar el destino que se encuentra en el AS1.

En este escenario, utilizando la implementación de BGP modificada, se puede dar el caso de no obtener todas las rutas en cada uno de los *peers*, esto puede deberse a los valores de algunos atributos de desempate de BGP (*MED*, *local preference*,...). Las diferencias que podrían suceder en la ejecución de este escenario se apreciarán a partir del apartado c de la figura 5.16. Para facilitar la comprensión de este estado final indeseado utilizaremos una nueva representación del intercambio de mensajes únicamente representando los últimos pasos del mecanismo. En la figura 5.17 se puede observar como en el apartado c el AS4 decide fijar como mejor ruta principal (3-1) y (2-1) como ruta secundaria. En el siguiente apartado (ap. d) se observa como AS6 tiene en sus listas de prefijos 3 posibles rutas: (4-3-1), (5-3-1) y (*4-2-1*), decidiendo tomar como ruta principal (4-3-1) se puede apreciar como ninguna de las rutas restantes son disjuntas a esta, por lo tanto, no podrá fijar un ruta secundaria ni enviar los mensajes *update* correspondientes a sus vecinos. Algo parecido le sucede al AS5 en el apartado e de esa misma figura, donde vemos que posee como ruta principal (3-1) y ninguna de las restantes (6-4-3-1) y (*3-4-2-1*) son disjuntas a esta. Por lo tanto en el estado final de las listas de prefijos de cada uno de los *peers* se observa que AS5 y AS6 no han encontrado ninguna ruta secundaria disjunta a la principal que han fijado. Este posible estado final indeseado hace que se denomine a esta topología como trampa.

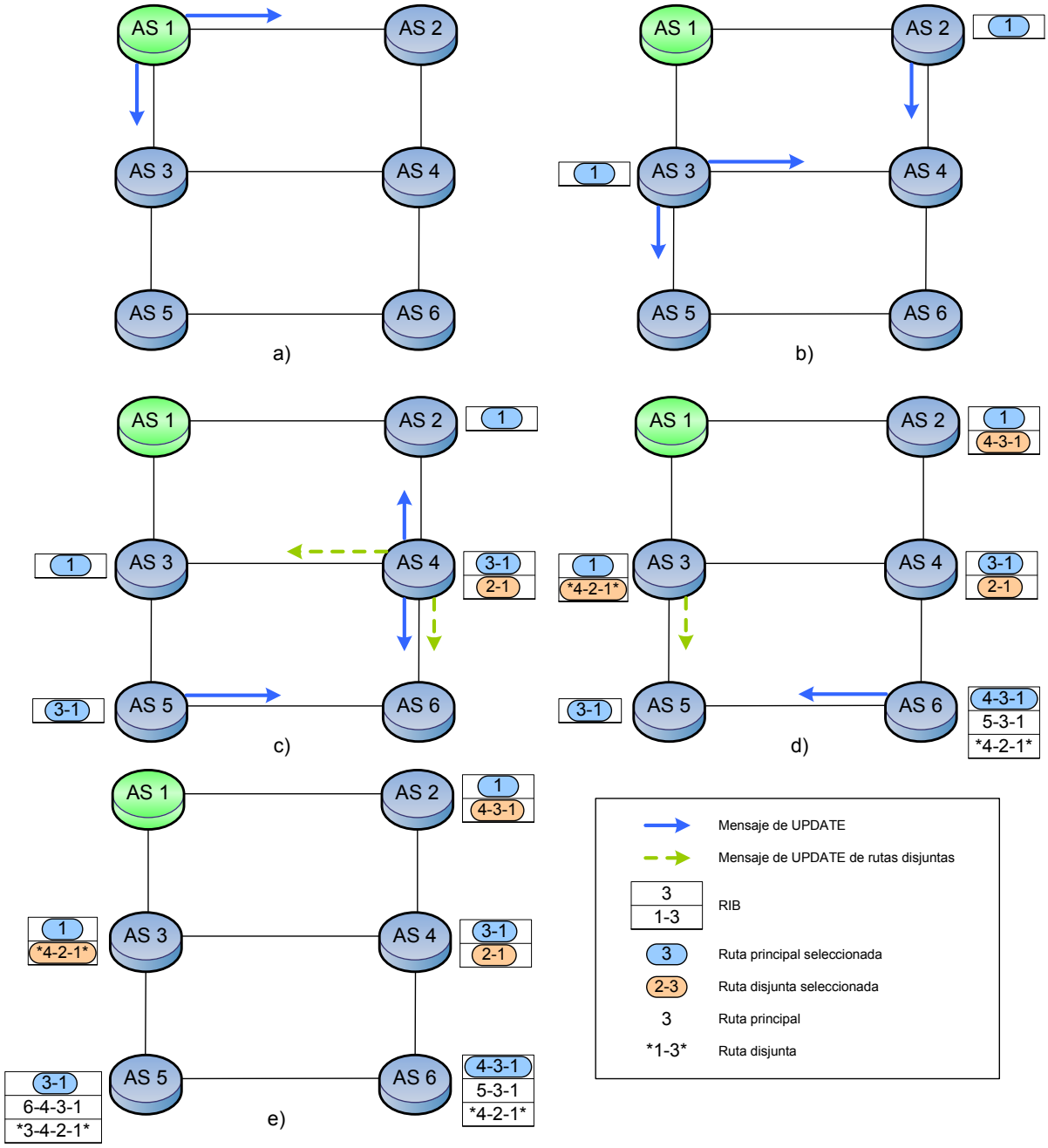


Figura 5.17: Escenario 4: Final indeseado de la propagación de rutas con BGP modificado.

En resumen, el problema que presenta esta topología es que dependiendo de la ruta que se fije como principal, será posible establecer una secundaria o no. Véase la lista de prefijos del AS6

al final de la simulación utilizando el protocolo BGP modificado: (4-3-1), (5-3-1) y (*4-2-1*). Como sucede en este caso, si utiliza la ruta (4-3-1) como principal, ninguna de las restantes es disjunta a ella, pero si fijara como principal la (5-3-1) se observa que la ruta (*4-2-1*) si es una ruta secundaria disjunta. En este tipo de topologías se observa como, a pesar de poseer rutas disjuntas en su *Adj-RIB-In*, determinados *peers* no han sido capaces de obtener ese par de rutas.

Una solución para el problema que presentan este tipo de topologías se propuso en el capítulo 7 de [1] y cuya explicación más detallada y posible implementación se propondrá en el último apartado de este proyecto "Trabajos futuros".

5.4.2. Resultados

A continuación se presentarán los resultados obtenidos tras la simulación del cuarto escenario. Cabe recordar que este escenario presentaba una topología que se denominaba "trampa" debido a la posibilidad que la misma presentaba de llegar a un estado final en el que algunos de los *peers* de la red no son capaces de obtener una ruta secundaria disjunta a la principal, teniendo en su lista de prefijos al menos dos rutas disjuntas entre sí. Como se viene realizando en todas las simulaciones de este proyecto, en primer lugar se presenta la figura 5.18 en la que aparece la topología de esta red acompañada de las direcciones de cada sub-red e interfaces de las mismas.

En todos los escenarios que se han simulado en este proyecto se ha utilizado *Wireshark* para realizar la captura de paquetes BGP así como las tablas de prefijos de la implementación de VNUML para obtener la información de las RIBs de cada uno de los *peers* de la red. Como ya se ha comentado anteriormente, tanto las capturas como las tablas de prefijos se encuentra almacenadas en el disco adjunto para su análisis si se considerara oportuno.

Se utilizará de aquí en adelante el mismo formato de tablas que se utilizó en el escenario anterior para visualizar el estado final de las listas de prefijos de cada uno de los *peers*. En las siguientes tablas se pueden observar los valores obtenidos al final de la simulación.

Para el desarrollo de la simulación de este escenario y el posterior cálculo de tiempos de ejecución, se han simulado todos los enlaces de la red con los mismos atributos (*MED*, *local*

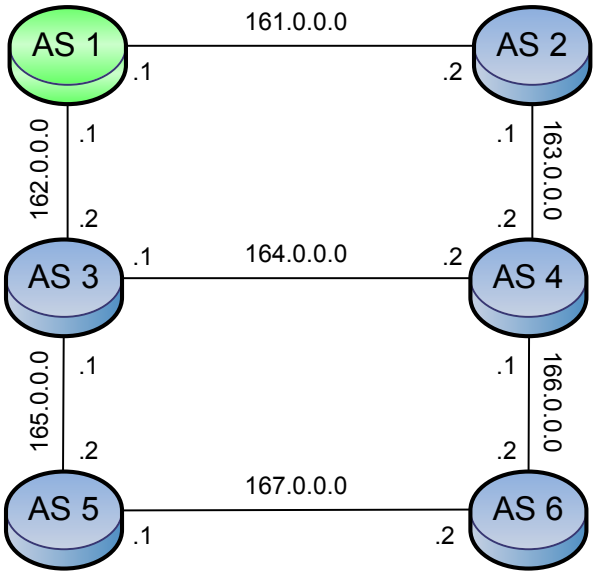


Figura 5.18: Escenario 4: Topología de red con direcciones.

preference,...), por lo que no se alcanzará el estado indeseado y por el contrario todos los *peers* de la red obtendrán una ruta principal y su correspondiente ruta secundaria disjunta en todas las iteraciones de la simulación. De esta forma se simplificarán los valores obtenidos en las siguientes tablas.

Destino	Tipo de ruta	Siguiente salto	Ruta	Princ.	Secund.
103.0.0.0	P	163.0.0.2	1	*	
	S	161.0.0.1	4 - 3 - 1		*

Tabla 5.21: Escenario 4: *Adj-RIB-In* del AS2.

Destino	Tipo de ruta	Siguiente salto	Ruta	Princ.	Secund.
103.0.0.0	P	162.0.0.1	1	*	
	P	164.0.0.2	4 - 2 - 1		*
	S	165.0.0.2	5 - 6 - 4 - 2 - 1		

Tabla 5.22: Escenario 4: *Adj-RIB-In* del AS3.

Destino	Tipo de ruta	Siguiente salto	Ruta	Princ.	Secund.
103.0.0.0	P	163.0.0.1	2 - 1	*	
	P	164.0.0.1	3 - 1		*
	S	166.0.0.2	6 - 5 - 3 - 1		

Tabla 5.23: Escenario 4: *Adj-RIB-In* del AS4.

Destino	Tipo de ruta	Siguiente salto	Ruta	Princ.	Secund.
103.0.0.0	P	165.0.0.1	3 - 1	*	
	S	165.0.0.1	3 - 4 - 2 - 1		
	P	167.0.0.2	6 - 4 - 2 - 1		*

Tabla 5.24: Escenario 4: *Adj-RIB-In* del AS5.

Destino	Tipo de ruta	Siguiente salto	Ruta	Princ.	Secund.
103.0.0.0	P	166.0.0.1	4 - 2 - 1	*	
	P	167.0.0.1	5 - 3 - 1		*
	S	166.0.0.1	4 - 3 - 1		

Tabla 5.25: Escenario 4: *Adj-RIB-In* del AS6.

En las tablas 5.21, 5.22, 5.23, 5.24 y 5.25 se puede comprobar como las entradas de cada una de ellas se corresponden con los valores obtenidos en la simulación teórica de este escenario que se ha analizado en la figura 5.16, en la que se presentaba el funcionamiento correcto del mecanismo de propagación de rutas de este escenario. De la misma forma que se ha realizado en el resto de escenarios, no se ha mostrado la tabla de prefijos del *peer* AS1 por tratarse del Sistema Autónomo en el que se encuentra la red destino.

De la misma forma que en los escenarios anteriores, se han realizado 20 simulaciones de esta topología y se ha realizado el estudio estadístico de intervalos de confianza para verificar la veracidad de los resultados obtenidos.

Los tiempos obtenidos para cada una de las simulaciones se pueden consultar en el fichero adjunto *TiemposEscenario4.pdf* que se encuentra en el directorio `/simulaciones/tiempos/` del cd adjunto.

Protocolo BGP utilizado	Media	Desviación típica
Normal	32,555	1,9
Modificado	62,0012	2,39431

Tabla 5.26: Escenario 4: Tiempo medio y desviación típica.

En la tabla 5.26 se pueden observar las medias y desviaciones típicas obtenidas para el escenario 4 en la utilización del protocolo BGP normal y modificado. Antes de analizar los resultados obtenidos se realizará el intervalo de confianza de los mismos para verificar la fiabilidad de los resultados. También habrá que tener en cuenta que, en estas simulaciones, se ha fijado la misma preferencia a todos los enlaces por lo que no aparecerá el problema de la topología "trampa" que se ha comentado anteriormente, es decir, todas las simulaciones que se han ejecutado alcanzaron el estado final deseable.

Si se calculan los intervalos de confianza de estas 20 simulaciones para cada implementación de BGP con un nivel de confianza del 95 %, se obtienen los intervalos presentados en la tabla 5.27.

BGP utilizado	Incremento obtenido	Intervalo obtenido
Normal	0,831 = 2,5526 %	32,555 ± 0,831
Modificado	1,0493 = 1,69239 %	62,0012 ± 1,0493

Tabla 5.27: Escenario 4: Intervalos de confianza obtenidos.

Como se observa en la tabla 5.27 tanto para BGP normal como para el modificado, tras realizar las simulaciones se resuelve que, con un 95 % de confianza, sus valores estarán comprendidos entorno a la media con un error entorno al 2 % del valor de la media en cada uno de los casos. Este resultado nos indica que los valores de tiempo de ejecución obtenidos tras realizar las simulaciones son válidos y podrán ser utilizados en las conclusiones pertinentes.

En la tabla 5.28 se puede observar los tiempos de ejecución de cada implementación de BGP. De la misma forma que en los dos escenarios anteriores se observa un incremento de unos 30

segundos que se analizará a continuación.

	Tiempo de ejecución (seg)
BGP original	32,5550
BGP modificado	62,0012

Tabla 5.28: Escenario 4: Tiempos de ejecución.

De la misma forma que sucedía en los escenarios anteriores (escenario 2 y 3), este incremento de 30 segundos en el tiempo de ejecución se debe al temporizador que se ha mencionado anteriormente. Para comprobar la situación que provoca este incremento de tiempo se debe consultar de nuevo la figura 5.16.

- En el paso c de la figura 5.16 se puede observar como el AS4 encuentra al mismo tiempo su ruta principal y secundaria. En ese momento debe notificar a su vecino AS6 ambas rutas, pero debido a la existencia del temporizador tendrá que esperar esos 30 segundos entre un envío y otro.

Al igual que sucedía en el escenario anterior, se ha incrementado la complejidad de la red con respecto a los escenarios más sencillos, de 3 y 4 *peers*, y a pesar de ello se observa como el tiempo de ejecución del mecanismo de propagación de rutas implementado únicamente sufre el retardo provocado por la existencia del temporizador de actualización de rutas. Este hecho vuelve a demostrar que la ejecución en segundo plano que se ha implementado para la propagación de rutas secundarias de dicho mecanismo no produce un retraso sustancial en la obtención de rutas principales ni secundarias disjuntas.

5.5. ESCENARIO 5

En este quinto escenario se realizará la simulación de un modelo de red algo más últil que los anteriores desde el punto de vista de similitud con las redes "reales". Se trata de un modelo práctico de la Red Paneuropea de Telecomunicaciones de fibra óptica que conecta los centros de negocios más importantes de Europa, entre los cuales se encuentran algunas ciudades españolas como Madrid, Barcelona o Valencia. En la figura 5.19 se pueden observar los distintos nodos y enlaces de la red que existe en la actualidad.

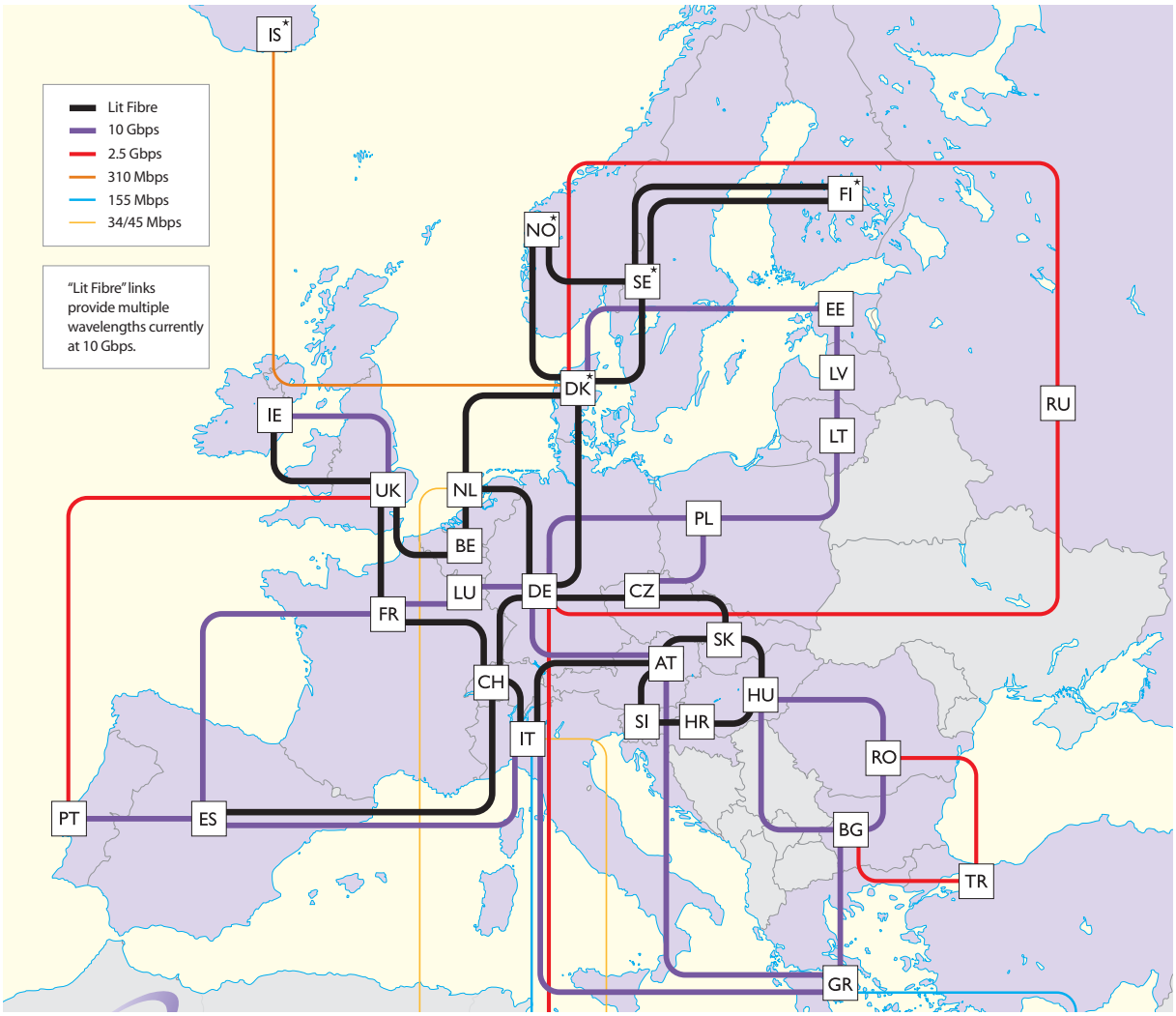


Figura 5.19: Esquema de la Red Paneuropea real.

Para poder llevar a cabo la simulación de este escenario tan complejo se han tenido que agrupar los nodos de la red para simplificar las conexiones y reducir su tamaño. Esta reducción de tamaño es necesaria si se tiene en cuenta que cada máquina virtual utilizada en este proyecto necesita un espacio aproximado de 1 gigabyte para su simulación dentro de la red. Finalmente, ante esta situación, se ha decidido agrupar todos los nodos de una determinada nación en un único Sistema Autónomo que englobará la red de ese país.

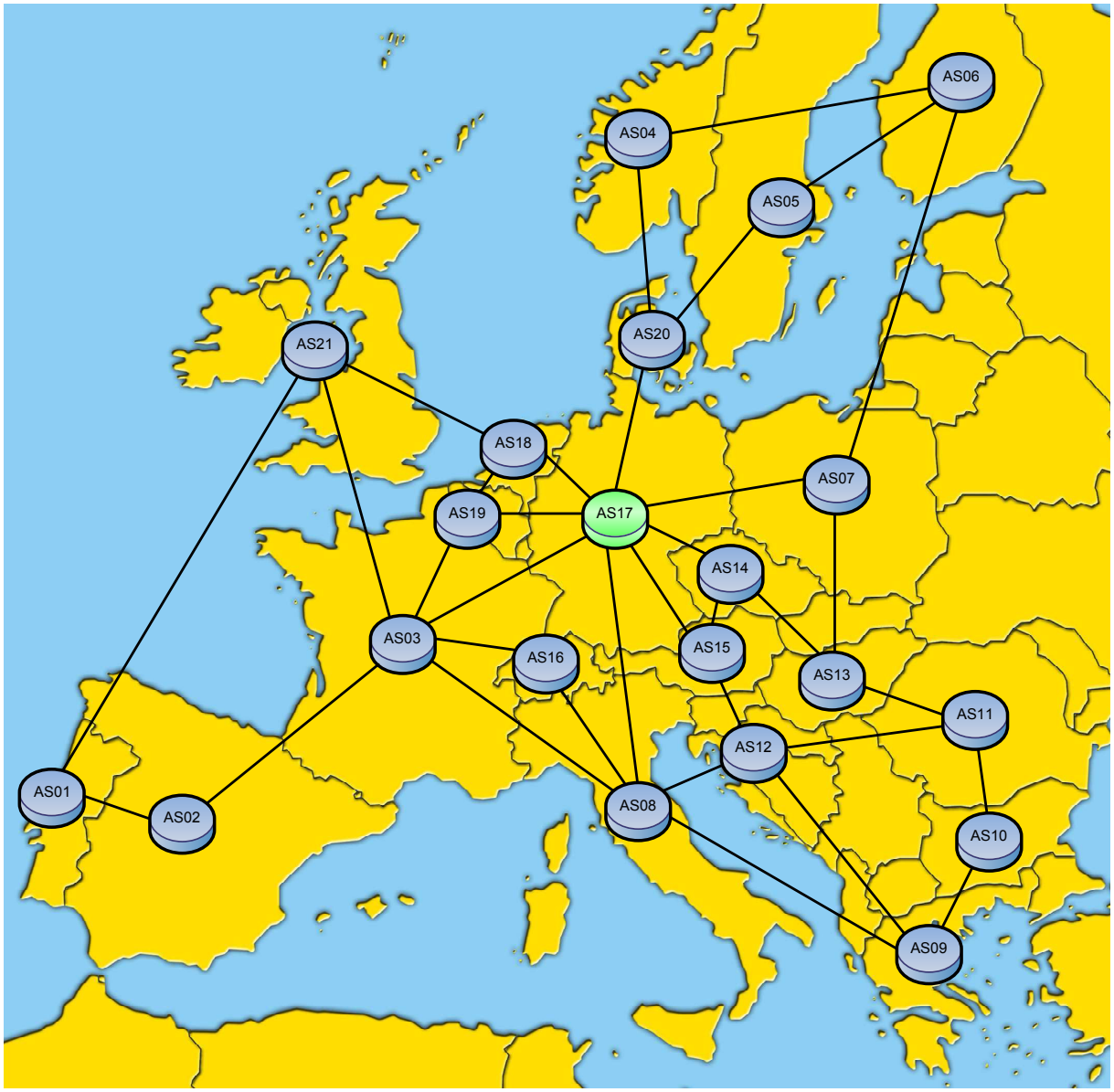


Figura 5.20: Escenario 5: Esquema de Red Paneuropea simulada.

En la figura 5.20 se puede observar que finalmente se utilizarán 21 *peers*. Uno de los enlaces que no se ha podido desarrollar en este modelo es el existente entre Dublín, Londres y Glasgow ya que su existencia provoca que la red no sea 2-conectada. En la propuesta de modificación del mecanismo de obtención de rutas de BGP se explicó detalladamente la necesidad de cumplir esta condición, red 2-conectada [2], para que el cálculo de la ruta secundaria disjunta que se propuso en [1] sea posible.

En esta simulación se debe tener en cuenta que el destino para el cual se realizará el mecanismo de propagación de rutas de BGP se encuentra en el AS17 que es Alemania. Se ha tomado este *peer* ya que se trata del más importante de la red que con 8 enlaces es el *peer* que mayor número de vecinos de la red posee.

5.5.1. Funcionamiento

Como se viene realizando en los apartados anteriores, se llevará a cabo un análisis previo del funcionamiento de este escenario mediante la utilización del protocolo BGP original. En este caso no se realizará un análisis gráfico del intercambio de mensajes *update* paso a paso como en los apartados anteriores, ya que esta red es mucho más compleja y el mecanismo ha quedado bastante claro con las indicaciones de los cuatro escenarios anteriores.

El inicio del intercambio de mensajes *update* de este escenario lo lleva a cabo AS17 (Alemania) que se encarga de notificar a sus vecinos la existencia de una red destino en su Sistema Autónomo. Y tras una serie de intercambios de mensajes *update* y modificaciones en las *Adj-RIB-In*, *Adj-RIB-Out* y *Loc-RIB* se alcanzará un estado final estable en cada uno de los *peers*. En la figura 5.21 se puede observar el estado final de las RIBs de cada uno de los *peers* de esta red, a excepción de AS17 que no tendrá ninguna entrada ya que el destino está en su propio Sistema Autónomo.

[2] Red que presenta, al menos, dos posibles caminos entre cualquier par de nodos origen y destino de la misma.

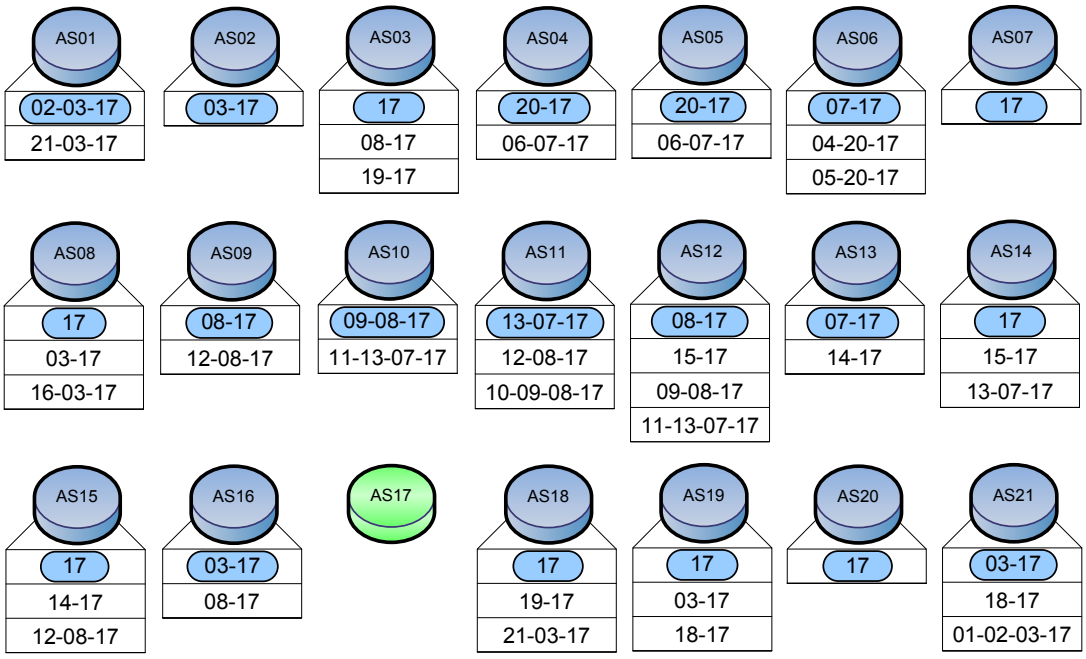


Figura 5.21: Escenario 5: RIB´s obtenidas con BGP original.

De la misma forma que se acaba de mencionar para BGP normal, en la figura 5.22 se muestra el estado final de las RIBs de los diferentes *peers* de la red utilizando el protocolo BGP modificado en este proyecto. Únicamente se debe destacar de estas RIBs que, como se puede observar, todos los *peers* de la red han seleccionado una ruta principal y otra secundaria disjunta para alcanzar el destino que se encuentra en AS17.

5.5.2. Resultados

A continuación se presentarán los resultados obtenidos tras la simulación del quinto escenario que consta de 21 *peers* por lo que se considerará la simulación más realista que se realizará en este proyecto. Como se ha explicado anteriormente, esta topología simulada es una simplificación de la red paneuropea que comunica los dominios de diferentes paises de Europa en el "mundo real".

En todos los escenarios que se han simulado en este proyecto se ha utilizado *Wireshark* para realizar la captura de paquetes BGP así como las tablas de prefijos de la implementación de VNUML para obtener la información de las RIBs de cada uno de los *peers* de la red. Como ya se

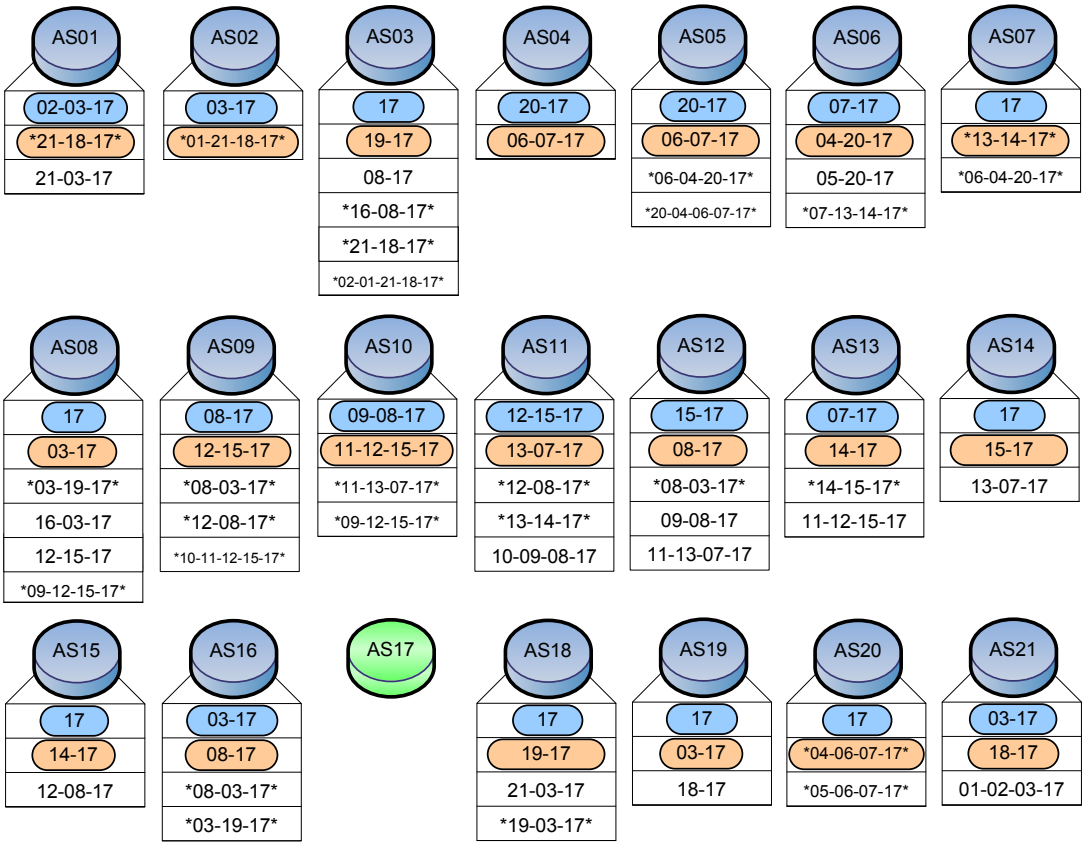


Figura 5.22: Escenario 5: RIB´s obtenidas con BGP modificado.

ha comentado anteriormente, tanto los tiempos, RIBs y mensajes *update* enviados se encuentran en el directorio `/simulaciones/` del disco adjunto que se entrega con este proyecto. Los ficheros correspondientes a este escenario son:

- Tiempos de ejecución: `/simulaciones/tiempos/TiemposEscenario5.pdf`.
- RIBs: en `/simulaciones/RIBs/` se encuentran los ficheros `RIB_escenario5normal.txt` y `RIB_escenario5disjuntas.txt`
- Captura de mensajes *update*: en `/simulaciones/capturas/` están `escenario5normal` y `escenario5disjuntas`.

Además de estos ficheros, en el apéndice E se muestran las tablas de prefijos obtenidas y detalladas con direccciones IP para la utilización de BGP normal y modificado.

A la hora de simular los escenarios previos se han realizado 20 simulaciones pero en este caso la complejidad del mismo y el número de máquinas virtuales que se deben ejecutar a la vez provoca una serie de problemas en las simulaciones que impiden su realización con normalidad. Al ejecutar las 21 máquinas virtuales su inicialización se realizó con normalidad pero a la hora de ejecutar el demonio BGP algunos equipos lo hacían con normalidad pero había otros que paraban el proceso sin motivo aparente. Al analizar este problema mediante la realización de múltiples sismulaciones se dedujo que el problema residía en el virtualizador VNUML debido a que en cada una de las simulaciones fallaba un *peer* diferente e incluso había ocasiones en la que todos funcionaban correctamente. Estos últimos casos son los que se han guardado en los ficheros comentados anteriormente y que, por tanto, se utilizarán para el análisis estadísticos mediante intervalos de confianza que se analizará a continuación.

Protocolo BGP utilizado	Media	Desviación típica
Normal	59,9962	6,09
Modificado	89,3594	1,0026

Tabla 5.29: Escenario 5: Tiempo medio y desviación típica.

En la tabla 5.29 se pueden observar las medias y desviaciones típicas obtenidas para el escenario 5 en la utilización del protocolo BGP normal y modificado. Antes de analizar los resultados obtenidos se realizará el intervalo de confianza de los mismos para verificar la fiabilidad de los resultados.

Si se calculan los intervalos de confianza de estas 5 simulaciones para cada implementación de BGP con un nivel de confianza del 95 %, se obtienen los intervalos presentados en la tabla 5.19. Se sabe que estos resultados no son del todo fiables debido al escaso número de simulaciones correctas que se han podido realizar.

Como se observa en la tabla 5.30 tanto para BGP normal como para el modificado, tras realizar las simulaciones se resuelve que, con un 95 % de confianza, sus valores estarán comprendidos entorno a la media con un error del 4,45 % del valor de la media para BGP normal y entorno a un 0,5 % para BGP modificado. Este resultado nos indica que los valores de tiempo de ejecución

BGP utilizado	Incremento obtenido	Intervalo obtenido
Normal	2,6701 = 4,45045 %	59,9962 ± 2,6701
Modificado	0,4394 = 0,4917 %	89,3594 ± 1,0026

Tabla 5.30: Escenario 5: Intervalos de confianza obtenidos.

obtenidos para cada caso tras realizar las simulaciones son muy próximos entre sí y pueden ser utilizados sin problemas, teniendo precaución con ellos debido a la escasez de resultados.

En la tabla 5.31 se puede observar los tiempos de ejecución de cada implementación de BGP. De nuevo se observa un incremento entorno a los 30 segundos como ha ocurrido en los escenarios anteriores.

	Tiempo de ejecución (seg)
BGP original	59,9962
BGP modificado	89,3594

Tabla 5.31: Escenario 5: Tiempos de ejecución.

De la misma forma que sucedía en los apartados anteriores, este incremento de 30 segundos en el tiempo de ejecución puede deberse al temporizador *MinRouteAdvertisementInterval* que utiliza BGP para fijar el tiempo entre actualizaciones de rutas para un mismo destino y un mismo *peer* vecino. Pero para poder afirmar este hecho habrá que comprobarlo con un análisis de la topología.

Al igual que sucedía en los otros escenarios esta situación mencionada en la que un *peer* desea enviar varias rutas para un mismo destino a un mismo vecino provoca que dicho *peer* tenga que esperar el temporizador mencionado entre esos envíos. Esta situación problemática en este caso no se podrá analizar con exactitud, pero observando la red deternidamente se pueden observar varios puntos conflictivos:

- AS09 a AS10: debe enviarle su ruta principal (09-08-17) y su secundaria (*09-12-15-17*).
- AS06 a AS05: debe enviarle su ruta principal (06-07-17) y su secundaira (*06-04-20-17*).
- AS20 a AS05: debe enviarle su ruta principal (20-17) y su secundaira (*20-04-06-20-17*).

Si se analiza la topología y las RIBs obtenidas con mayor detenimiento pueden aparecer más situaciones de este tipo. Finalmente se puede concluir algo muy similar a lo sucedido en los escenarios anteriores: la ejecución de las nuevas funciones de la nueva implementación de BGP no suponen un coste de tiempo excesivo con respecto al tiempo de ejecución de BGP normal. Como se puede observar en la comparativa de tiempos, existe una diferencia de tiempo de 30 segundos provocada por el temporizador de BGP por lo que parece que la nueva implementación de BGP no incrementa este tiempo gracias a su ejecución en segundo plano.

5.6. COMPARATIVA

A la vista de los resultados obtenidos en cada uno de los escenarios propuestos, tabla 5.32, se pueden extraer varias conclusiones relevantes sobre cada uno de los escenarios simulados así como la influencia del número de nodos sobre el tiempo de ejecución obtenido.

En primer lugar se puede observar que el único escenario que prácticamente mantiene su tiempo de ejecución es el escenario 1, esto se debe a que dicho escenario no necesita enviar ningún mensaje *update* de ruta secundaria lo cual hace que el tiempo de distribución de rutas principal y secundaria sea casi el mismo que en el caso de utilizar BGP normal.

Por otro lado, en los escenarios restantes (2, 3, 4 y 5) se observa una diferencia "curiosa" ya que la diferencia entre los tiempos de ejecución utilizando BGP normal y modificado en todos los casos es de unos 30 segundos más o menos. Esta situación hizo que se investigara con mayor detalle el funcionamiento de BGP hasta descubrir el mecanismo de BGP que provocaba este retraso tan particular: el temporizador *MinRouteAdvertisementInterval*, que evita que un *peer* envíe dos rutas consecutivas para un mismo destino a un mismo vecino, es decir, cuando un *peer* de cualquier escenario desea enviar su ruta principal y disjunta a un mismo *peer* vecino, estos envíos deben esperar 30 segundos (valor del temporizador por defecto) entre cada uno de los envíos.

Entonces, tras explicar el retardo provocado por dicho temporizador hay que tener en cuenta que este nuevo mecanismo de propagación de rutas secundarias que se ha implementado en este proyecto se ejecuta de en paralelo al mecanismo de rutas de BGP normal. Debido a este hecho,

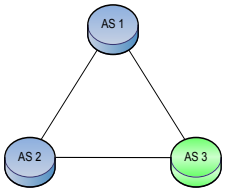
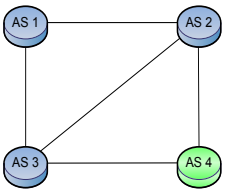
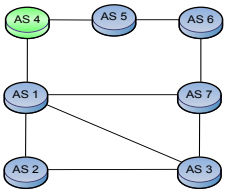
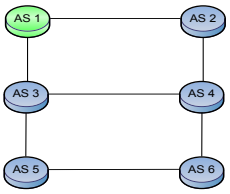
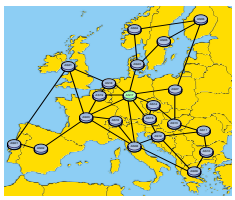
RESULTADOS DE LAS SIMULACIONES	
Escenario 1 	BGP Original: 27,1484 seg BGP Modificado: 28,9118 seg
Escenario 2 	BGP Original: 30,8140 seg BGP Modificado: 59,7725 seg
Escenario 3 	BGP Original: 60,2936 seg BGP Modificado: 92,1794 seg
Escenario 4 	BGP Original: 32,5550 seg BGP Modificado: 62,0012 seg
Escenario 5 	BGP Original: 59,9962 seg BGP Modificado: 89,3594 seg

Tabla 5.32: Comparativa de los resultados obtenidos para los escenarios propuestos.

se puede concluir que el tiempo de ejecución de la mejora implementada es prácticamente despreciable frente al tiempo total de ejecución del mecanismo de propagación de rutas.

También se ha podido comprobar la variación de este tiempo de ejecución en función del número de nodos de la red, se puede observar como, tanto para redes de 4 *peers* (escenario 2) como para redes de 21 nodos (escenario 5), el retardo que provoca la ejecución en segundo plano del mecanismo de propagación de rutas secundarias disjuntas implementado es muy bajo con respecto al total.

Capítulo 6

CONCLUSIONES

En este proyecto se ha desarrollado la implementación del mecanismo de obtención de caminos (AS_PATHs) disjuntos extremo a extremo en BGP basado en la solución propuesta en el capítulo 7 de [1]. Tras analizar las limitaciones que presenta el funcionamiento "normal" del protocolo BGP y realizar las modificaciones correspondientes sobre el código C del software de enrutamiento quagga se alcanzó una nueva versión de dicho protocolo de enrutamiento que que ha sido probada en las máquinas virtuales de los diferentes escenarios propuestos para verificar el correcto funcionamiento de las modificaciones realizadas.

Como se ha explicado a lo largo de esta memoria, para realizar las pruebas de estos escenarios se decidió utilizar la herramienta de virtualización de redes VNUML, que a pesar de las dificultades y complejidad que presenta a la hora de realizar una simulación, ofrece unos resultados muy próximos a la realidad así como una interfaz muy cómoda para el usuario que parece dar acceso a una red totalmente real permitiendo ejecutar los escenarios realizados en este documento.

En el capítulo 5 de este proyecto "Simulaciones y resultados" se han estudiado diferentes escenarios en orden creciente de complejidad de sus respectivas redes y, por consiguiente, mayor dificultad para su simulación y comprobación de resultados. Como se ha podido observar en el capítulo anterior, se comenzó simulando una red muy simple con sólo 3 *peers* que no necesitaba enviar ningún mensaje *update* de ruta secundaria y el último escenario consta de 21 *peers* con múltiples intercambios de rutas principales y secundarias hasta alcanzar el estado final en el que todos los *peers* de la red obtuvieran AS_PATHs disjuntos extremo a extremo.

	Tiempo medio para BGP normal (seg)	Tiempo medio para BGP modificado (seg)	Diferencia media obtenida (seg)
Escenario 1	27,1484	28,9118	1,7634
Escenario 2	30,8140	59,7725	28,9585
Escenario 3	60,2936	92,1794	31,8858
Escenario 4	32,5550	62,0012	29,4462
Escenario 5	59,9962	89,3594	29,3632

Tabla 6.1: Tiempos medios de ejecución y diferencias.

En la tabla 6.1 se pueden observar los tiempos medios de ejecución obtenidos para cada uno de los escenarios y versiones del protocolo BGP utilizadas: se tiene una columna con los resultados del protocolo BGP normal y otra para la versión modificada. Por último en la columna de la derecha se muestra la diferencia de esos tiempos medios obtenidos cuyos valores se analizarán a continuación.

La primera conclusión que se puede obtener a la vista de los resultados obtenidos en la tabla 6.1 es la diferencia de unos 30 segundos que se obtiene en todos los escenarios excepto en el primero. Como ya se explicó en cada uno de los casos, esta diferencia de tiempo se debe a la existencia del temporizador *MinRouteAdvertisementInterval* que se encarga de regular el envío de mensajes *update* para un mismo destino a un *peer* vecino en concreto. Por lo tanto, en las simulaciones realizadas en este proyecto, cuando un *peer* pretende enviar su ruta principal para un destino determinado a un *peer* vecino y posteriormente enviarle, a ese mismo *peer*, la ruta secundaria disjunta obtenida, deberá esperar esos 30 segundos para poder realizar el envío. Esta espera se produce porque en la implementación que se ha desarrollado en este proyecto tanto las rutas principales como secundarias se envían contenidas en los mismos mensajes *update*, únicamente modificando un "flag" que indica el tipo de ruta enviada en ese mensaje. Posteriormente, al no haber modificado las condiciones de reseteo del temporizador, este interpreta que se están enviando dos actualizaciones consecutivas de una misma ruta por lo que se deben esperar esos 30 segundos para poder realizar ese envío de rutas. En el último capítulo de este documento, trabajos futuros, se ha propuesto la reducción del valor del temporizador. Esto mejoraría el ren-

dimiento del protocolo a priori pero no es una modificación trivial, es decir, podría perjudicar al funcionamiento habitual de BGP. Para realizar esta mejora habría que analizar con detenimiento las consecuencias que la misma provoca.

Como se ha comentado anteriormente, el primer escenario no sufre ese incremento de tiempo medio de 30 segundos puesto que no debe enviar ningún mensaje *update* de ruta secundaria a ninguno de sus vecinos y, por lo tanto, no se verá afectado por el efecto del temporizador. Sin embargo, este primer escenario, sufre un incremento de tiempo medio de 1,7634 segundos que supone un 6,5 % de incremento del tiempo medio de ejecución con respecto al protocolo BGP normal. Este incremento tan bajo se debe a la ejecución en paralelo que lleva a cabo el mecanismo de propagación de rutas secundarias que se ha implementado. El funcionamiento es el siguiente: tras enviar las primeras rutas principales alguno de los *peers* obtendrá la primera ruta secundaria de la red. En ese momento ese *peer* envía la notificación de ruta secundaria a todos sus vecinos, excepto al siguiente salto de la ruta disjunta seleccionada, y al mismo tiempo se siguen distribuyendo las rutas principales mediante el mecanismo de propagación de rutas habitual de BGP. Por este motivo se dice que la ejecución del mecanismo implementado se realiza en segundo plano o en paralelo. En el resto de escenarios se puede observar que los únicos retardos que sufren los tiempos de ejecución de BGP son los referidos a los 30 segundos del temporizador *MinRouteAdvertisementInterval* que se ha comentado anteriormente.

Para realizar un análisis más completo del funcionamiento de la red y de las prestaciones del mecanismo de obtención de AS_PATHs implementado hubiera sido útil calcular la carga computacional de cada una de los *peers* de cada una de las topologías. Este valor no ha podido ser obtenido mediante la herramienta de simulación utilizada, VNUML, puesto que al lanzar las diferentes máquinas virtuales de la red, en la carga computacional de cada una de ellas se tendrá una parte de la propia máquina virtual y otra referida a la carga utilizada por el núcleo del sistema anfitrión. Para obtener correctamente este parámetro de las simulaciones habría que utilizar una herramienta *software* adecuada para este propósito como podría ser *Virtual Center*. El objetivo principal de este proyecto ha sido implementar el nuevo mecanismo de obtención de rutas de BGP comprobando el incremento de tiempo de ejecución que producía esta modificación por lo que, finalmente, se decidió no llevar a cabo la obtención de la carga computacional por no ser un factor determinante en este caso.

Finalmente, se ha alcanzado el objetivo principal del proyecto que era desarrollar la implementación del mecanismo de obtención de AS_PATHs disjuntos extremo a extremo en BGP y realizar las respectivas simulaciones que verifican su correcto funcionamiento para varias topologías diferentes. También se han tenido en cuenta las consideraciones que se realizaron en el capítulo 3 del actual documento: sencillez, viabilidad y escalabilidad. En primer lugar se puede decir que la solución que se ha implementado es sencilla puesto que se han utilizado los mismos mensajes *update* del protocolo original modificando únicamente un *flag* y de la misma forma una serie de modificaciones sobre el código de quagga que prácticamente no han modificado su estructura. La viabilidad de esta implementación se ha demostrado en la realización de simulaciones correctas en tiempos razonables. Por último la escalabilidad queda plasmada en la simulación del escenario 5, una red con 21 *peers* muy próxima a la realidad, donde se han obtenido tiempos cercanos al funcionamiento de BGP original, sin tener en cuenta el retardo provocado por el temporizador. A la vista de estas acciones y resultados se puede decir que se han alcanzado los objetivos fijados previamente.

Capítulo 7

TRABAJOS FUTUROS

En este apartado se presentarán algunas de las futuras líneas de trabajo que pueden realizarse sobre la versión ya implementada que ha resultado tras las modificaciones introducidas a lo largo de este proyecto fin de carrera.

7.1. Modificación del mecanismo de selección de rutas

En primer lugar se propone una modificación del funcionamiento del mecanismo de selección de rutas secundarias que ya se ha mencionado en la simulación del escenario 4 de este mismo proyecto. En ese apartado se explicaba el problema que tienen algunas topologías concretas a la hora de seleccionar sus rutas principal y secundaria, se vió como puede darse el caso de tener una lista de prefijos con dos o más rutas disjuntas pero que dependiendo de la ruta principal seleccionada sería posible o no la selección de una ruta secundaria disjunta.

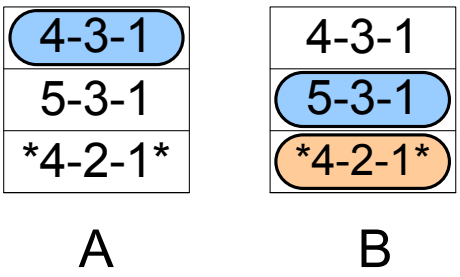
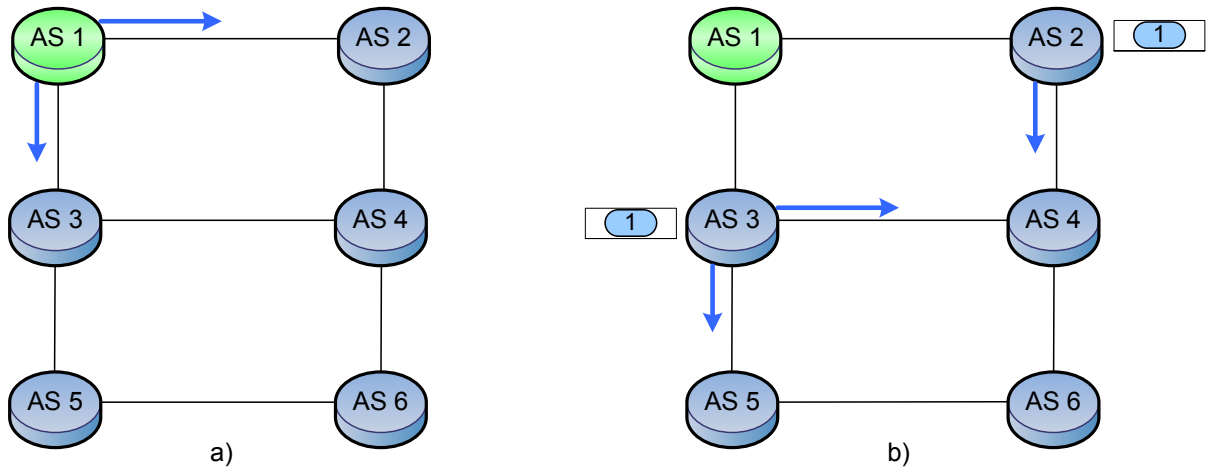


Figura 7.1: *Adj-RIB-In* del AS6 del escenario 4.

En la figura 7.1 se observan los dos posibles estados finales de la selección de rutas principal y secundaria del AS6 del escenario 4 simulado en este mismo proyecto. En el estado A se observa que la ruta seleccionada como principal es la (4-3-1) impidiendo esto la posibilidad de establecer una ruta secundaria. Sin embargo, en el estado B se puede ver como la ruta seleccionada como principal ha sido la (5-3-1) permitiendo este hecho fijar la ruta secundaria disjunta como (*4-2-1*), obteniendo así el resultado deseado para la simulación de cualquier escenario, es decir, todos los *peers* de la red conocen una ruta principal y secundaria para alcanzar un destino determinado.

La modificación que habría que realizar sobre la implementación del protocolo BGP que se ha desarrollado en este proyecto tendría lugar en la función de selección de rutas. En la selección de las rutas secundarias se comprueba la correcta elección de la misma y habría que introducir una condición que, en caso de que no fuera posible encontrar una ruta secundaria disjunta, se debe disminuir el grado de preferencia de la ruta principal actual, de modo que al volver a ejecutar el mecanismo de selección sea elegida la otra ruta principal existente. De igual forma, volvería a ejecutar el mecanismo de selección de ruta secundaria y así sucesivamente. Al recibir una nueva ruta secundaria se debe comprobar si la ruta anterior es válida para restituir su grado de preferencia.

A continuación se presenta la figura 7.2[1] en la que se observa como deberían ser el intercambio de mensajes y el estado de las listas de prefijos en caso de implementarse esta mejora.



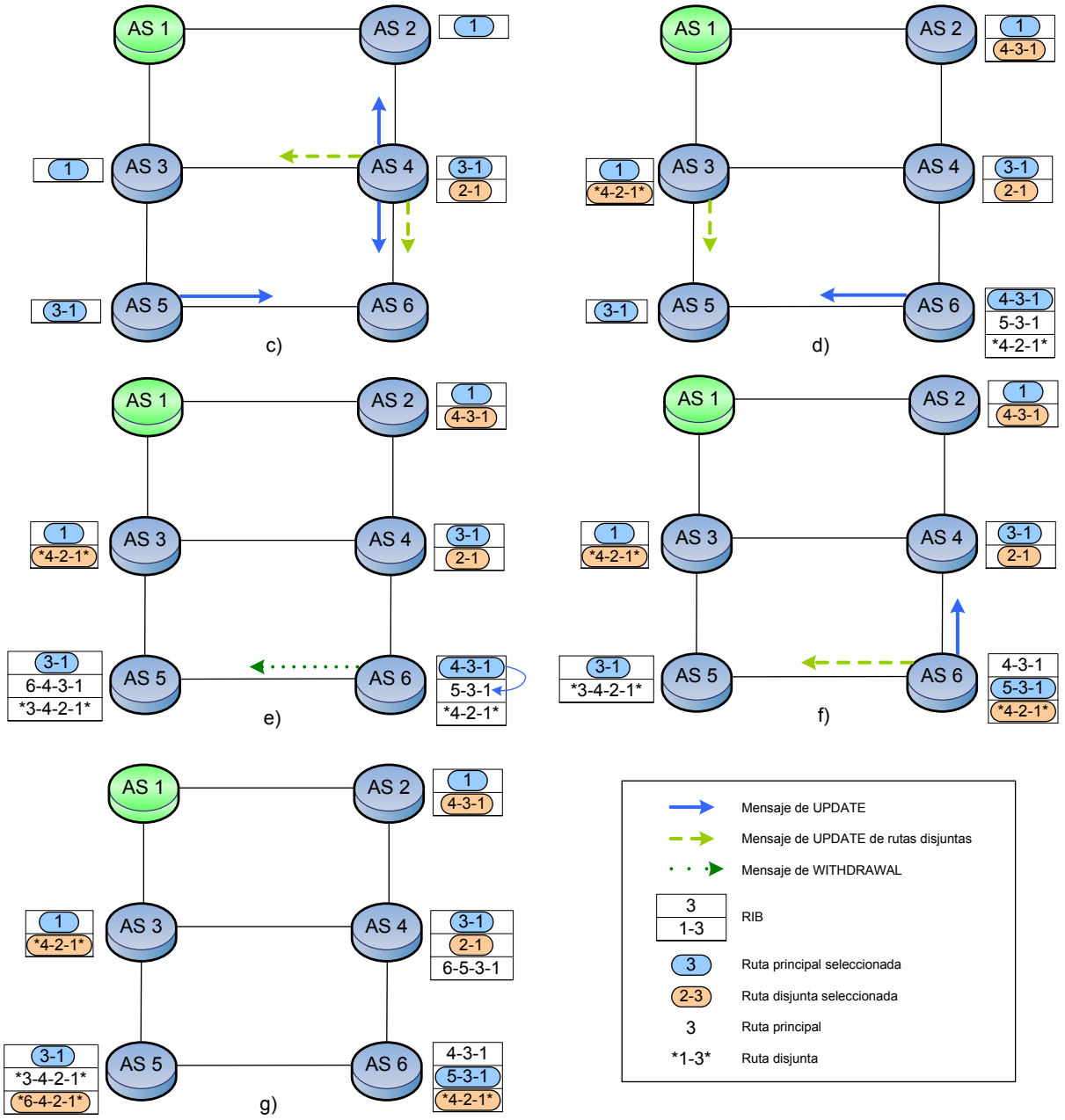


Figura 7.2: Escenario 4: Propagación de rutas con mejora de BGP.

Se puede observar en la figura 7.2 como en el apartado e, el AS6 reduce el grado de preferencia de la ruta principal (4-3-1) y vuelve a ejecutar el algoritmo de selección de rutas, quedando elegida (5-3-1) como mejor ruta principal. Por este motivo se envía el mensaje *withdrawal* a AS5, que aparece en esa misma figura. Posteriormente, al haber fijado AS6 una nueva ruta principal

y como consecuencia de ello la secundaria (*4-2-1*)(ap. e) se enviarán a sus respectivos vecinos AS4 y AS5 los mensajes *update* de ruta principal y secundaria respectivamente. Por último se puede ver en el apartado f de la misma figura como AS5 también ha fijado su ruta secundaria (*6-4-2-1*) y por lo tanto, todos los *peers* de la red han obtenido sus respectivas rutas principal y secundaria disjunta.

En esta primera línea futura de trabajo se ha presentado una modificación que permitiría al protocolo BGP evitar que la selección de la ruta principal sea una "trampa" para la que no hay ruta disjunta (aún existiendo dos rutas disjuntas entre sí).

7.2. Reseteo del temporizador de actualización de rutas

A lo largo de este proyecto se han implementado una serie de cambios con el objetivo de obtener rutas secundarias disjuntas a la principal, en el apartado de *Simulaciones y resultados* se han desarrollado diferentes escenarios con el objetivo común de analizar el tiempo de ejecución de este mecanismo de propagación de rutas mencionado.

En cada uno de los escenarios estudiados se analizó el tiempo de ejecución, observando que siempre se veía incrementado pero llamaba especialmente la atención el aumento en los escenarios 2, 3 y 4, en los que dicho incremento de tiempo era cercano a los 30 segundos. Como ya se explicó en cada uno de estos apartados, este tiempo extra que necesitaba la nueva implementación del mecanismo para propagar las rutas, se debía al temporizador que utiliza BGP[1] para enviar los anuncios de actualización de rutas. En el apartado 9.2.3.1 de [6] se explica con mayor detalle como existe un parámetro en BGP denominado *MinRouteAdvertisementInterval* que determina el tiempo mínimo que debe transcurrir entre el envío de dos mensajes de anuncio de rutas desde un *peer* a otro vecino para cada destino posible. La versión 4.0 de BGP que se ha utilizado en este proyecto fija por defecto este tiempo a 30 segundos.

[1] Se puede obtener más información sobre los temporizadores de BGP en el apartado 6.4 de los apéndices de [6].

Para comprender mejor el problema que se podría resolver en este futuro trabajo se utilizará la simulación realizada en el escenario 2 de este proyecto que a continuación se recuerda. Para el segundo escenario implementado se tenían 4 *peers* contectados como aparecía en la figura 5.6 y la propagación de rutas principales y secundarias que tenía lugar se pudo observar también en la figura 5.8.

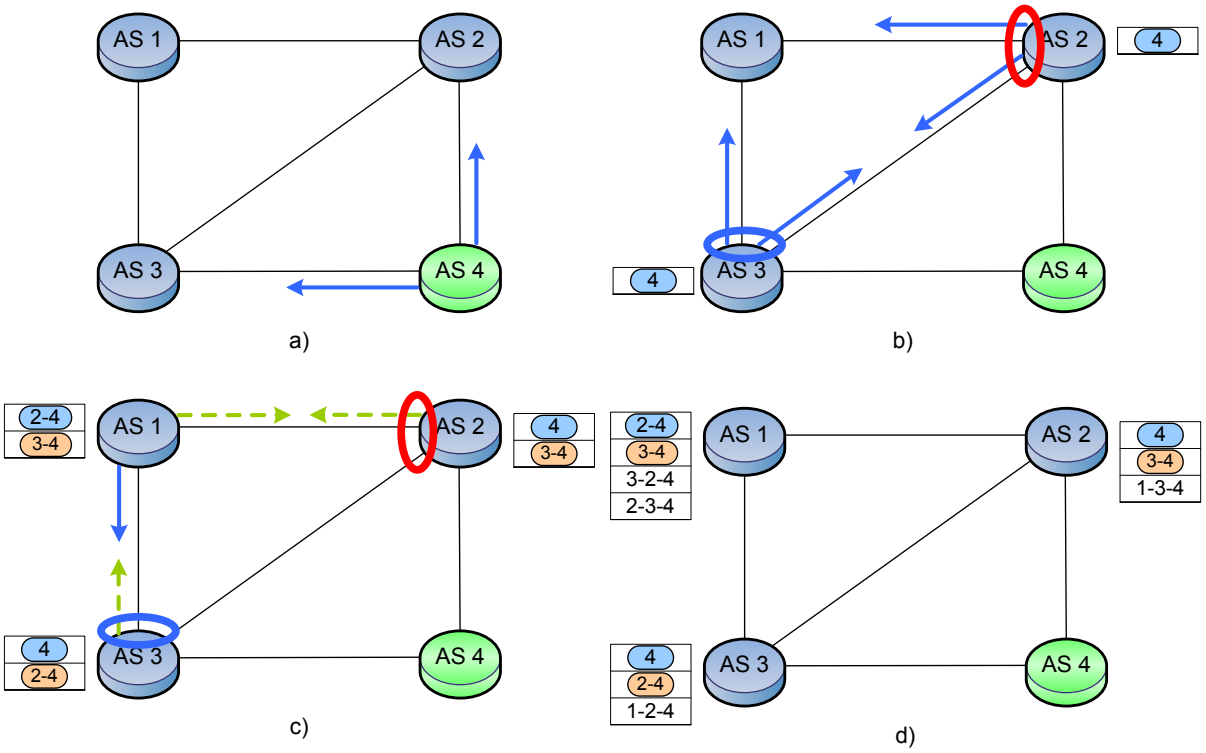


Figura 7.3: Momento conflictivo del mecanismo en el escenario 2.

En la figura 7.3 se puede observar de nuevo este intercambio de mensajes *update* para distribuir las rutas sobre un destino en AS4, pero en este caso aparece marcado el instante conflictivo que se propondrá solucionar en esta línea futura. En este caso el error se produce por partida doble, en primer lugar se observa marcado en color rojo que AS2 envía a AS1 su nueva ruta principal(ap. b), posteriormente envía de nuevo a AS1 la ruta secundaria aprendida(ap. c). Como el protocolo BGP únicamente permite el envío de actualización de rutas cada 30 segundos, AS2 tendrá que esperar esos 30 segundos para poder enviar su ruta secundaria. Esto se debe a que

en la implementación que aquí se ha desarrollado se han utilizado los mismos mensajes *update* tanto para rutas principales como secundarias, únicamente modificando el valor de un bit, por lo que BGP "cree" que se está anunciando una nueva ruta para el mismo destino y por ello debe esperar ese temporizador.

En la figura 7.3 también se puede observar, marcados en azul claro, el instante del error producido para AS3 de manera similar a lo comentado en el párrafo anterior para AS2.

La solución que se podría realizar en este trabajo futuro sería llevar a cabo el reseteo del valor de este temporizador de actualización de rutas en el momento que un AS intente enviar seguidas dos rutas de distinto tipo, es decir, se tendrían los siguientes casos posibles, teniendo siempre en cuenta que se envían rutas para un mismo destino y *peer* vecino:

1. Envío de dos rutas del mismo tipo (principales o secundarias)
   Tendría que esperar 30 segundos entre ambos envíos.

2. Envío de ruta principal y secundaria (o viceversa):
   Se enviaría la ruta principal, al intentar enviar la ruta secundaria se resetearía el valor del temporizador e inmediatamente se enviaría la ruta secundaria.

En la RFC de BGP[6] se explica con mayor detalle la utilización de este temporizador y se puede observar que posee un complejo mecanismo para su actualización y funcionamiento, de manera que no es trivial la realización de esta modificación propuesta ni la posible reducción del valor del tiempo de dicho temporizador. Para desarrollar estas modificaciones correctamente habría que estudiar con mayor detenimiento los cambios que éstas provocarían sobre el funcionamiento de BGP y verificar estas mejoras con las simulaciones pertinentes.

7.3. Creación de *clusters* de dominios de AS

Sucede normalmente que entre los diferentes ASes de una red existen acuerdos particulares que definen el sistema de intercambio de tráfico, tránsito, *peering* o *multihoming*. Es evidente que los acuerdos a este nivel pueden ser de gran utilidad y es muy probable que el propio mercado define como se materializará.

La modificación que se podría realizar sobre este modelo de colaboración sería muy parecido al actual. En este modelo sólo los ASes con un acuerdo podrán finalizar túneles interdominio en los routers de otro dominio. Siguiendo este modelo se podría aplicar el mismo funcionamiento a los túneles de respaldo que se han obtenido en la implementación de este proyecto. Sólo aquellos ASes con acuerdos podrán enviar tráfico de respaldo por ellos. Se formarían pequeñas "islas" de ASes con acuerdos, en las que algunos de los ASes pueden formar parte de varias "islas", haciendo de tránsito en los respaldos.

En el ejemplo que aparece representado en la figura 7.4 se tienen dos "islas" de ASes que tienen acuerdos entre sí. Los AS4 y AS5 forman parte de ambas "islas". En esta figura se representan las rutas principales y secundaria para alcanzar un destino en AS8. Se observa como los ASes de la "isla B" tienen ambas rutas en sus RIBs, mientras que los ASes de la "isla A" únicamente poseen una ruta principal. En el caso de AS1 se observa como posee una ruta principal para cualquier *peer* de la red y una secundaria para cada *peer* de su "isla".

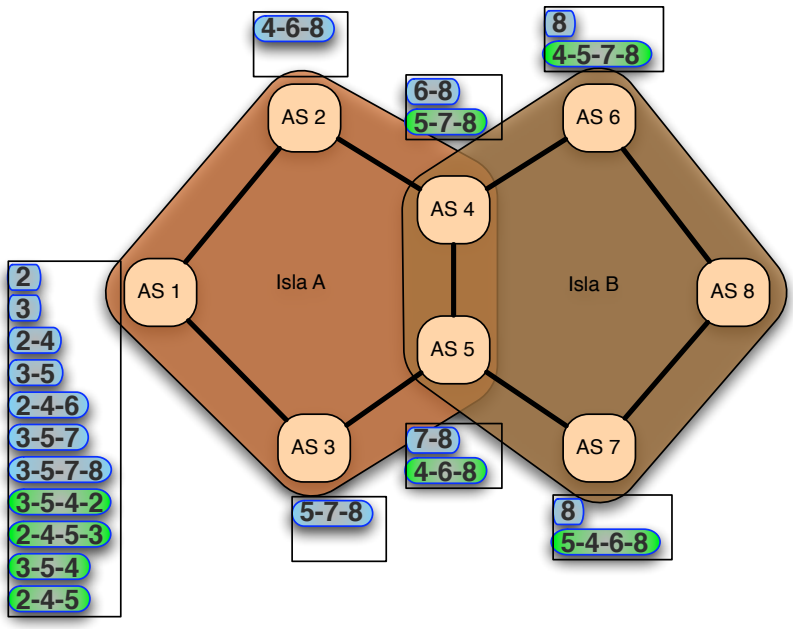


Figura 7.4: Grupos de Sistemas Autónomos con acuerdos distintos.

El aumento de las rutas secundarias en las RIBs y la convergencia para obtener esas rutas secundarias es la correspondiente a 5 dominios en cada "isla", tamaño y tiempo despreciable

frente al de manejar inmensas tablas BGP de cualquier dominio.

A continuación se explicará, con ayuda de un ejemplo representativo, que incluso en el caso de que un Sistema Autónomo completo dejara de funcionar, no se perdería la información de respaldo que no pertenece a la propia "isla". Se supone que el tráfico que AS1 desee enviar a AS8 atravesará la ruta (3-5-7-8). Como entre el AS1 y AS8 no hay acuerdo alguno, AS1 encaminará su tráfico a través del LSP abierto con AS5.

Si tuviera lugar un fallo en el enlace entre AS5 y AS8 sería AS5 quien se encargaría de encaminar el tráfico hacia AS8 por la nueva ruta (3-5-4-6-8), donde (4-6-8) es la ruta secundaria de respaldo de AS5 para llegar a AS8.

Si se produjera un fallo en el propio AS5, entonces lo primero que haría AS1 sería enviar su tráfico hasta AS5 por su ruta secundaria de respaldo (2-4-5). Pero si el fallo hubiera sucedido en todo el AS5 entonces AS1 recibirá una nueva notificación de fallo por la ruta secundaria. Esto le puede hacer sospechar que el fallo no sea en el enlace, sino en todo el AS5. Como no se puede hacer nada con el tráfico que pasa por el AS5 habrá que reencaminar los mensajes desde el *peer* anterior al que produjo el error, es decir, AS4. Entonces AS1 encaminará su tráfico por su ruta principal hasta AS4 (2-4) y AS4 se encargará de alcanzar el destino en AS8 a través de su ruta principal (2-4-6-8), ya que su ruta secundaria (2-4-5-7-8) pasa por el AS5 y el AS4 ya sabe que ese Sistema Autónomo está fallando.

Este problema únicamente sucede cuando un AS que forma parte de dos "islas" falla (AS5), entonces se reencamina el tráfico a partir del nodo anterior (AS4), pero esta solución no tiene porque funcionar siempre. La mejor solución a este problema sería que todos los *peers* de una "isla" conocieran cuales de los ASes con los que tienen acuerdos pertenecen a ambas "islas".

En cualquier caso, las rutas secundarias o de respaldo se utilizan en cuanto se detecta un fallo en la red. Después del mismo, la utilización de esta ruta secundaria puede prolongarse hasta que el fallo es arreglado, hasta que las nuevas rutas BGP son estables y se pueden volver a utilizar o en cualquier momento que se desee para optimizar la utilización de la red. La decisión la tomará el operador de la misma.

APÉNDICES

APÉNDICE A

Configuración de un router para VNUML

En este apéndice se analizará la configuración del fichero `bgpd.conf` de uno de los *peer* de una red básica para comprender la configuración que se lleva a cabo en ellos para lanzar el demonio BGP dentro de las máquinas virtuales que son, posteriormente, simuladas con ayuda de la herramienta de virtualización VNUML que ya ha sido mencionada en el estado del arte.

Para que la comprensión de este código sea lo más clara posible, en la figura B.1 se muestra la estructura de la red que se está configurando en este caso en concreto. En este apéndice se centrará la explicación en el fichero de configuración de un *peer* de AS4.

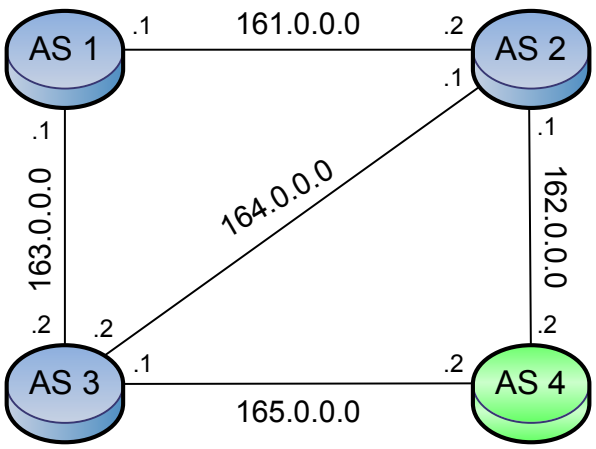


Figura A.1: Red para ejemplo de fichero de configuración.

Además de tener presente las conexiones de la red (AS4 tiene como *peers* vecinos a AS2 y AS3), en el cuadro siguiente se muestra el código necesario para la configuración del *peer* de AS4.

Las 10 primeras líneas de todos los ficheros de configuración que se han utilizado para la simulación de los escenarios de este proyecto son las mismas y se muestran a continuación:

```
# Configuration for router 'R4'
!
log file /tmp/cuatro.debug
!
debug bgp
debug bgp events
debug bgp filters
debug bgp fsm
debug bgp keepalives
debug bgp updates
```

Como se puede observar en la tercera línea se indica la ubicación del fichero de registro (`/tmp/cuatro.debug`) y desde la 5 a la 10 se configuran los mensajes del protocolo BGP que se quieren almacenar en el fichero de registro indicado anteriormente. Para el desarrollo de este proyecto es importante destacar que se guardará la información referente a los eventos, mensajes *update* y *keepalive* que se produzcan durante las simulaciones.

```
hostname R4
password bgp
!
router bgp 65004
```

En estas líneas de código se configura el nombre del router "R4" que se está configurando y la contraseña necesaria para acceder a dicho al mismo y poder visualizar durante la simulación el estado de las tablas de enrutamiento, la lista de prefijos aprendidos, etc. También se puede ver que en la tercera línea se configura a R4 como un *router* BGP, perteneciente al AS4 y se le asigna a este sistema autónomo el valor 65004. Éste valor será utilizado en el resto de ficheros de configuración para referenciar al *router* de este sistema autónomo.

Las conexiones que han sido establecidas con R4 deben ser configuradas para un correcto intercambio de prefijos en el protocolo BGP. Está configuración de la conexión de sus interfaces se realiza en las siguientes líneas:

```
network 103.0.0.0/16

neighbor 162.0.0.1  remote-as 65002
neighbor 162.0.0.1  soft-reconfiguration inbound
neighbor 162.0.0.1  prefix-list ANY out
neighbor 162.0.0.1  prefix-list 65002-PREFIXES in

neighbor 165.0.0.1  remote-as 65003
neighbor 165.0.0.1  soft-reconfiguration inbound
neighbor 165.0.0.1  prefix-list ANY out
neighbor 165.0.0.1  prefix-list 65003-PREFIXES in
```

Se analizará el contenido de esta porción de código dónde se pueden ver tres partes muy bien diferenciadas:

- Existencia de una red interna en AS4 (103.0.0.0).
- *Router* de AS2 es vecino de AS4.
- *Router* de AS3 es vecino de AS4.

Se observa tanto en el caso del vecino AS2 como AS3 que al configurar su conexión BGP se establecen unos claros criterios de aceptación/denegación de prefijos, que son las siguientes: cualquier *router* de la red, a través de un enlace en concreto, envía todos los prefijos que haya recibido (`prefix-list ANY out`) y únicamente acepta los prefijos procedentes de su vecino en este enlace (`prefix-list 65002-PREFIXES in`). A modo de ejemplo, en la interfaz 162.0.0.1 de AS4, le mandará a AS2 todos los prefijos que aprenda por cualquiera de sus interfaces y únicamente aceptará de éste sus prefijos propios.

La última parte del fichero de configuración consta de las definiciones de los prefijos iniciales que tendrá en su tabla interna cada uno de los sistemas autónomos y serán los prefijos que intercambiarán usando el protocolo BGP.

```
ip prefix-list MY-PREFIXES description Allow AS 65004 prefixes
ip prefix-list MY-PREFIXES seq 10 permit 162.0.0.0/16 le 32
ip prefix-list MY-PREFIXES seq 20 permit 165.0.0.0/16 le 32
ip prefix-list MY-PREFIXES seq 30 permit 103.0.0.0/16 le 32
!
ip prefix-list 65002-PREFIXES description Allow AS 65002 prefixes
ip prefix-list MY-PREFIXES seq 10 permit 161.0.0.0/16 le 32
ip prefix-list MY-PREFIXES seq 20 permit 162.0.0.0/16 le 32
ip prefix-list MY-PREFIXES seq 30 permit 164.0.0.0/16 le 32
!
ip prefix-list 65003-PREFIXES description Allow AS 65003 prefixes
ip prefix-list MY-PREFIXES seq 10 permit 163.0.0.0/16 le 32
ip prefix-list MY-PREFIXES seq 20 permit 164.0.0.0/16 le 32
ip prefix-list MY-PREFIXES seq 30 permit 165.0.0.0/16 le 32
```

Estos prefijos es únicamente un listado de las redes a las que tiene acceso cada uno de los ASes. En primer lugar vemos que aparece el listado de prefijos del propio AS4 y después la lista de cada uno de sus vecinos (AS2 y AS3). Como se ha hecho en el párrafo anterior, se centrará la explicación en un ejemplo para comprenderlo mejor. En la figura B.1 se puede observar que el *router* del AS2 está conectado a las redes 161.0.0.0, 162.0.0.0 y 164.0.0.0, de manera que AS2 tendrá que "permitir" la recepción de prefijos de estas subredes. Se puede apreciar que en las cuatro primeras líneas se realiza justamente esta acción.

Como se ha podido observar en este apéndice, la configuración de cada nuevo *router* de la red conlleva los siguientes pasos de configuración:

- Fichero de registro de la simulación.
- Dirección IP de las interfaces de vecinos y redes internas.
- Listado de prefijos propios y de cada vecino.

Éstos archivos de configuración serán cargados en el sistema de ficheros de cada una de las máquinas virtuales de la red a través de un comando `bgpd -d`. Este comando será ejecutado por un fichero de simulación *cuatro.xml*, analizado en el apéndice B, utilizando el virtualizador de redes VNUML.

APÉNDICE B

Escenario de simulación en VNUML

El fichero que se va a analizar en este apéndice se llama *cuatro.xml* y ha sido desarrollado para el escenario de la figura B.1, es el mismo que se utilizó en el apéndice A para explicar el funcionamiento de los ficheros de configuración de cada *router*. En este caso únicamente mostraremos los fragmentos de código más significativos, con el fin de simplificar al máximo la explicación ofrecida.

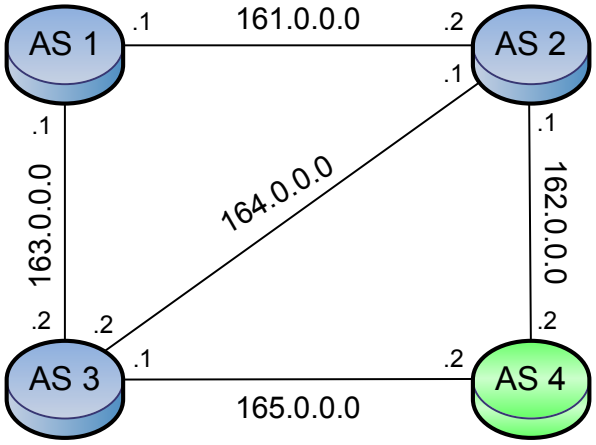


Figura B.1: Red ejemplo de escenario de simulación.

En primer lugar se deben llevar a cabo las definiciones globales del escenario de simulación a realizar, entre las cuales únicamente vamos a destacar las que aparecen a continuación:

```
<global>
   ...
   <simulation_name>Escenario II</simulation_name>
   ...
   <vm_defaults>
     <filesystem type="cow">&VNUMLDIR;filesystems/root_fs</filesystem>
     <kernel>&VNUMLDIR;kernels/linux</kernel>
     <console id="1">xterm</console>
   </vm_defaults>
</global>
```

- Nombre de la simulación: *"Escenario II"*
- Sistema de ficheros de cada router: `root\_fs`, este fichero ha sido obtenido de la página web del distribuidor oficial de VNUML[4]. Lo más importante de este sistema de ficheros utilizado es que incluye una implementación de BGP con el software de enrutamiento quagga instalado.
- Sistema operativo: linux, ya que es el sistema utilizado por el sistema de ficheros de que se dispone.
- Tipo de consola: xterm, será utilizada para acceder a cada *router* durante la simulación.

En el siguiente fragmento de código se observan las distintas redes existentes en el escenario a simular, en concreto tenemos 5 redes que conectan los 4 ASes del escenario.

```
<!-- Networks -->
<net name="AS1-AS2" mode="virtual_bridge"/>
<net name="AS1-AS3" mode="virtual_bridge"/>
<net name="AS2-AS3" mode="virtual_bridge"/>
<net name="AS2-AS4" mode="virtual_bridge"/>
<net name="AS3-AS4" mode="virtual_bridge"/>
```

Por último queda analizar la porción más importante de código, es decir, la configuración de las direcciones IP de las interfaces, así como la definición de los eventos que provocará el inicio o finalización del demonio BGP. El código es el siguiente:

```
<vm name="R4">

   <if id="1" net="AS2-AS4">
      <ipv4 mask="255.255.255.252">162.0.0.2</ipv4>
   </if>

   <if id="2" net="AS3-AS4">
      <ipv4 mask="255.255.255.252">165.0.0.2</ipv4>
   </if>
```

Se observa el código para la configuración de R4. Para definir los diferentes interfaces que tendrá dicho router es necesario configurar 4 valores para cada uno de ellos:

- Identificador: número natural (1,2,...) que diferencia cada interfaz.
- Red: identificador de la red a la que estará conectado el interfaz definido.
- Máscara de red a la que se conecta el interfaz.
- Dirección IP: del interfaz que se está configurando.

Tras asignar las interfaces del *router* se tendrán que configurar los eventos que llevará a cabo el *router* en caso de producirse un determinado evento. El código aparece en el cuadro siguiente:

```
   <filetree seq="start" root="/usr/local/etc">conf/R4</filetree>
   <exec seq="start" type="verbatim">hostname</exec>
   <exec seq="start" type="verbatim">rm -f /tmp/cuatro.debug</exec>
   <exec seq="start" type="verbatim">chmod -R 777 /usr/local/etc</exec>
   <exec seq="start" type="verbatim">&EXECDIR;bgpd -d</exec>
   <exec seq="stop" type="verbatim">hostname</exec>
   <exec seq="stop" type="verbatim">killall bgpd</exec>
```

Se puede ver que en caso de producirse el evento *start*:

- Copia el fichero *conf/R4* al sistema de ficheros local.
- Muestra el nombre del terminal que lanza el *start*.
- Elimina el fichero de registro existente (*cuatro.debug*).

- Establece permisos de lectura/escritura para la carpeta */usr/local/etc* que será donde se guarden los ficheros ejecutables de BGP.
- Activa el demonio BGP del terminal.

Y en caso de producirse el evento *stop*:

- Muestra el nombre del terminal que lanza el *stop*.
- "Mata" todos los procesos BGP que se estén ejecutando en este terminal.

El resto de terminales se configurarían de la misma forma dentro del mismo fichero de configuración del escenario de simulación *cuatro.xml*. Por último se hace notar que en esta configuración no aparecen las redes internas de cada sistema autónomo (p.e. 103.0.0.0 en AS4) ya que esa información únicamente es relevante para el intercambio de prefijos de BGP, por lo que solamente habría que indicar la existencia de la misma en sus ficheros de configuración (apéndice A).

APÉNDICE C

Ejecución de una simulación en VNUML

En este apéndice se mostrarán los pasos a seguir para ejecutar un escenario de simulación generado para el virtualizador VNUML. Se verán los comandos necesarios para iniciar el escenario, lanzar el demonio BGP y posteriormente parar la simulación.

En la figura C.1 se puede ver una red muy simple que se utilizará como hilo conductor de este breve tutorial.

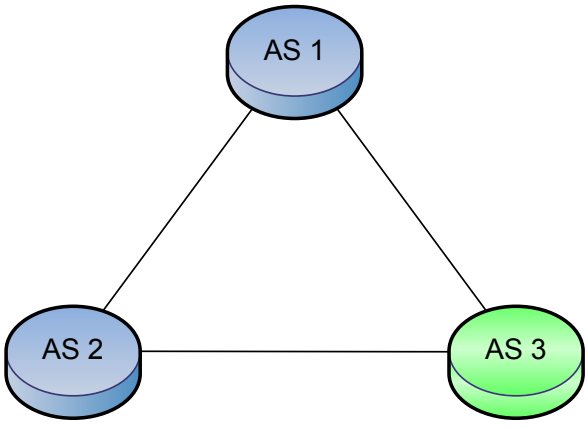


Figura C.1: Red ejemplo de ejecución de simulación.

C.1. Lanzamiento de la simulación

En la carpeta en la que se tenga guardado el escenario de simulación, en este caso será `tres.xml`. En primer lugar se deben adquirir permisos de administrador y posteriormente lanzar la simulación en VNUML. Los comandos son los siguientes:

```
sudo -s
vnumlparser.pl -t tres.xml -v -u root
```

C.2. Activación del demonio BGP

En el apéndice B se ha explicado como llevar a cabo la configuración de los eventos provocados por la ejecución del evento *start* que vemos a continuación:

```
vnumlparser.pl -x start@tres.xml -v
```

La activación del demonio BGP provocará el consiguiente intercambio de mensajes *open*, *update* y *keepalive*. Provocando unas modificaciones en la capacidad de enrutamiento de la red ejemplo.

C.3. Desactivación del demonio BGP

De la misma forma que se ha activado el demonio existe una orden, también configurada por el usuario, para desactivarlo y, por tanto, perder las rutas aprendidas durante su periodo de actividad. El comando a ejecutar sería:

```
vnumlparser.pl -x stop@tres.xml -v
```

C.4. Finalización de la simulación

De la misma forma que se ha lanzado la simulación, existe un comando para detenerla. Éste comando se encarga de parar la simulación así como de parar la ejecución del demonio BGP. El comando para realizar esta acción sería el siguiente:

```
vnumlparser.pl -d tres.xml -v -u root
```

C.5. Acceso a los *routers* de la simulación

Para acceder a cada uno de los *routes* de la simulación se utilizará siempre el usuario *root* y la contraseña *xxxx*. Una vez se ha accedido al terminal se pueden ejecutar una serie de comandos en función de la información que se quiera obtener:

Comando	Función
ps -e	Muestra los procesos en ejecución
ifconfig	Muestra los interfaces que tiene configurados
route -n	Muestra la tabla de rutas disponibles

Tabla C.1: Funciones del terminal.

C.6. Monitorización del protocolo BGP

Para poder comprobar el funcionamiento del protocolo BGP parece interesante tener alguna forma de visualizar las tablas de prefijos, los vecinos, los mensajes enviados, así como los valores de cada uno de los campos de éstos. Para llevar a cabo estas funciones habrá que iniciar una sesión *telnet* con el router que se desee monitorizar al puerto 2605, que es sobre el que se ejecuta BGP:

```
telnet R4 2605
```

La contraseña para acceder al puerto sobre el que se ejecuta BGP fue fijada en el fichero de configuración de cada *router*, tal y como vimos en el apéndice A, y siempre toma el valor *bgp*.

Una vez se haya accedido al puerto 2605 del *router* podremos ejecutar cualquiera de los siguiente comandos para monitorizar el estado del protocolo BGP en la red:

Comando	Función
show ip bgp	Muestra la tabla de prefijos de enrutamiento de BGP.
show ip bgp X.X.X.X	Muestra únicamente la tabla de prefijos de enrutamiento para la red indicada.
show ip bgp summary	Muestra el estado de todas las conexiones BGP.
show ip bgp neighbors	Muestra el número de rutas anunciadas/rebididas por este *peer*.

Tabla C.2: Funciones para la monitorización de BGP.

De cualquier forma, si fueran necesarias otras funciones, se puede obtener información detallada sobre ellas en el capítulo 9 del manual[5] de la página oficial de quagga.

APÉNDICE D

Modificación del código de BGP en quagga

En este apéndice se presentarán los pasos necesarios para realizar cualquier modificación que un usuario quiera llevar a cabo sobre el código de un programa instalado en un sistema de ficheros Linux. En concreto y en aplicación al desarrollo realizado en este proyecto se centrará esta explicación en el estudio de las modificaciones realizadas sobre el código C del protocolo BGP que posee quagga en su actual implementación. Se realizarán las modificaciones pertinentes, se recompilará el código de quagga y se ejecutará posteriormente cualquier simulación que utilice este sistema de ficheros Linux modificado.

D.1. Modificación de los mecanismos de BGP

En primer lugar habría que descargar el código de la versión de quagga que se desea modificar, en nuestro caso se ha utilizado la versión 0.98.6, es un software libre descargado de la web oficial de quagga[9]. Al descargar esta versión y descomprimir el archivo obtenido se obtienen una serie de carpetas con los distintos protocolos que soporta esta versión de quagga: BGP, OSPFv2, OSPFv3, RIP, RIPng, SNMP y Zebra. En este proyecto se prestará especial atención al contenido de la carpeta `/quagga-0.98.6/bgpd/` donde se encuentran los archivos .c y .h que contienen el código del funcionamiento del demonio BGP de quagga. Se puede obtener más información sobre la implementación de BGP en el manual del mismo[5].

Las modificaciones realizadas sobre estos ficheros de código C ya han sido comentadas con suficiente detalle en el apartado de implementación de este proyecto por lo que aquí simplemente se hará una breve mención de algunos de los ficheros susceptibles de ser modificados para realizar las modificaciones requeridas: `bgpd.c`, `bgpd.h`, `bgpd_advertise.c`, `bgpd_advertise.h`, `bgpd_route.c`, `bgpd_route.h`, entre otros.

D.2. Montaje del sistema de ficheros

Para poder modificar el código de quagga que está ejecutando el sistema de ficheros que se está utilizando en las simulaciones será necesario montarlo sobre nuestro equipo local para poder recompilar el código y obtener los nuevos ficheros ejecutables.

Para montar el sistema de ficheros en un directorio local se debe ejecutar los siguientes comandos:

```
sudo -s
mount -o loop /usr/share/vnuml/filesystems/root_fs_tutorial-0.6.0.copy /mnt
```

El primer comando sirve para adquirir permisos de administrador (`sudo -s`) y el segundo monta el sistema de ficheros Linux "root_fs_tutorial-0.6.0.copy" obtenido de [9] sobre el directorio local `/mnt/`. De esta forma se puede acceder a los ficheros ejecutables de quagga, en concreto a los de BGP, de manera local.

D.3. Recompilar el código de quagga

Para poder probar el efecto de las modificaciones realizadas sobre el código de BGP sobre quagga habrá que recompilar su código de manera que los nuevos ejecutables generados sean utilizados por el sistema de ficheros Linux al ejecutar las simulaciones. Los siguientes comandos muestran los pasos a seguir:

```
cd /home/apt/srodriguez/PFC/quagga-0.98.6/

./configure --disable-ipv6 --enable-static --prefix=/usr/local

make clean

make

make install

cp -R /usr/local/sbin/* /mnt/usr/local/sbin/
```

En los comandos anteriores se pueden observar las siguientes acciones realizadas:

1. Se accede a la carpeta donde se encuentra el código de quagga (`/quagga-0.98.6/`)
2. Se configuran las opciones de compilación de quagga: sin IPv6, estática y con destino `/usr/local`.
3. Se eliminan todos los ejecutables existentes de compilaciones anteriores.
4. Se vuelve a compilar el código de quagga.
5. Se instala la nueva versión de quagga modificado.
6. Por último se copian los ficheros del directorio local (`/usr/local/sbin/*`), donde han sido generados, al directorio pertinente del sistema de ficheros que se ha montado previamente (`/mnt/usr/local/sbin/`).

Nota importante: Antes de ejecutar una simulación con una nueva versión del código de quagga es muy importante eliminar todos los temporales de la ejecución anterior, ya que sino se realiza esta acción, la simulación confundiría el sistema de ficheros a utilizar y las direcciones de los ficheros ejecutables. El comando necesario para eliminar los temporales de la simulación es:

```
rm -rf /home/apt/srodriguez/.vnuml/*
```

Como se puede observar en la línea de comando anterior, `rm` es el comando *remove* que sirve para eliminar uno o varios archivos y las opciones `-rf` sirven para eliminar estos ficheros definitivamente y sin pedir confirmación al usuario. Por otro lado se puede ver que los temporales de la simulación son almacenados en el directorio local del usuario, en este caso `/home/apt/srodriguez/`, en un directorio oculto que recibe el nombre de `/.vnuml/`. Al escribir la opción * al final de la ruta indica que se desean eliminar todos los archivos de ese directorio.

D.4. Desmontaje del sistema de ficheros

Por último se procederá a desmontar el sistema de ficheros que se había montado para cargarle la nueva versión de quagga modificada. Para realizar esta acción bastará con ejecutar los siguientes comandos:

```
umount /mnt
```

Para desmontar el sistema de ficheros montado en `/mnt` únicamente habría que ejecutar el comando anterior. Pero si al ejecutarlo se produjera un error del tipo "*Device is busy*" significa que todavía hay algún proceso ejecutándose sobre ese sistema de ficheros, por lo que habría que cancelar esos procesos:

```
lsof -t /mnt

fuser -9 7234
```

Para cancelar algún proceso que se ejecute sobre el sistema de ficheros montado, en primer lugar ejecutamos el comando `lsof -t /mnt` que nos indicará el identificador del proceso que afecta a este directorio. Este indicador es un entero que para este ejemplo valdrá 7234. Una vez obtenido este valor se procede a cancelar la ejecución de ese proceso en concreto, el comando `fuser -9 7234` se encarga de parar el proceso con ese indicador permitiendome desmontar el sistema de ficheros previamente montado.

D.5. Ejecución de las nuevas simulaciones

Para ejecutar cualquier simulación con esta nueva versión del protocolo BGP modificado sobre quagga, únicamente se debe indicar al comienzo del fichero de configuración del escenario el sistema de ficheros que se quiere utilizar:

```
<filesystem>/usr/share/vnuml/filesystems/root_fs_tutorial-0.6.0</filesystem>
        ó
<filesystem>/usr/share/vnuml/filesystems/root_fs_tutorial-0.6.0.new</filesystem>
```

En las líneas anteriores se puede observar como en primer lugar la simulación está utilizando el sistema de ficheros original descargado de [9] cuyo nombre es `root_fs_tutorial-0.6.0` y en segundo lugar se observa que utilizaría el sistema de ficheros `root_fs_tutorial-0.6.0.new` que es una copia modificada del sistema de ficheros inicial con las modificaciones pertinentes del codigo de BGP sobre quagga.

Otra opción para ejecutar un nuevo sistema de ficheros es modificar directamente el que se ha obtenido de la web de quagga (`root_fs_tutorial-0.6.0`), únicamente se copian los nuevos archivos ejecutables de quagga sobre este sistema de ficheros y al volver a ejecutar la simulación se estará utilizando la nueva versión. En cualquier caso, se recomienda utilizar un nuevo sistema de ficheros renombrado para evitar errores de ejecución o pérdida de información al sobreescribir archivos ejecutables existentes.

APÉNDICE E

RIBs del escenario 5 simulado

En este apéndice se presentarán las tablas de prefijos obtenidas para el escenario 5 simulado en este proyecto en cada uno de los modos de funcionamiento de BGP, se analizará en primer lugar las RIBs obtenidas utilizando el protocolo BGP normal y posteriormente la nueva implementación modificada en este proyecto.

Además se debe recordar, como ya se ha mencionado en la memoria, que en el directorio `/simulaciones/` del disco adjunto a este proyecto se pueden encontrar las tablas de prefijos, tiempos de ejecución y captura de mensajes *update* de las simulaciones realizadas para este escenario tan complejo.

En la figura E.1 se puede observar como se ha etiquetado cada enlace entre *peers* con el valor de los 8 primeros bits de sus direcciones IP, de manera que el número de la etiqueta indica X.0.0.0, siendo X el valor que figura en cada una de las etiquetas. Para simplificar la representación lo máximo posible se ha decidido adoptar un convenio para la asignación de IP a las interfaces de cada *peer*: en cualquier enlace existente, el interfaz del Sistema Autónomo cuyo identificador sea mayor tendrá el valor .1 y el menor será .2. Por ejemplo, en el enlace AS01 - AS02 se tendrá que el interfaz de AS01 será 161.0.0.1 y el de AS2 161.0.0.2, siendo 161 el valor de la etiqueta que aparece en la figura.

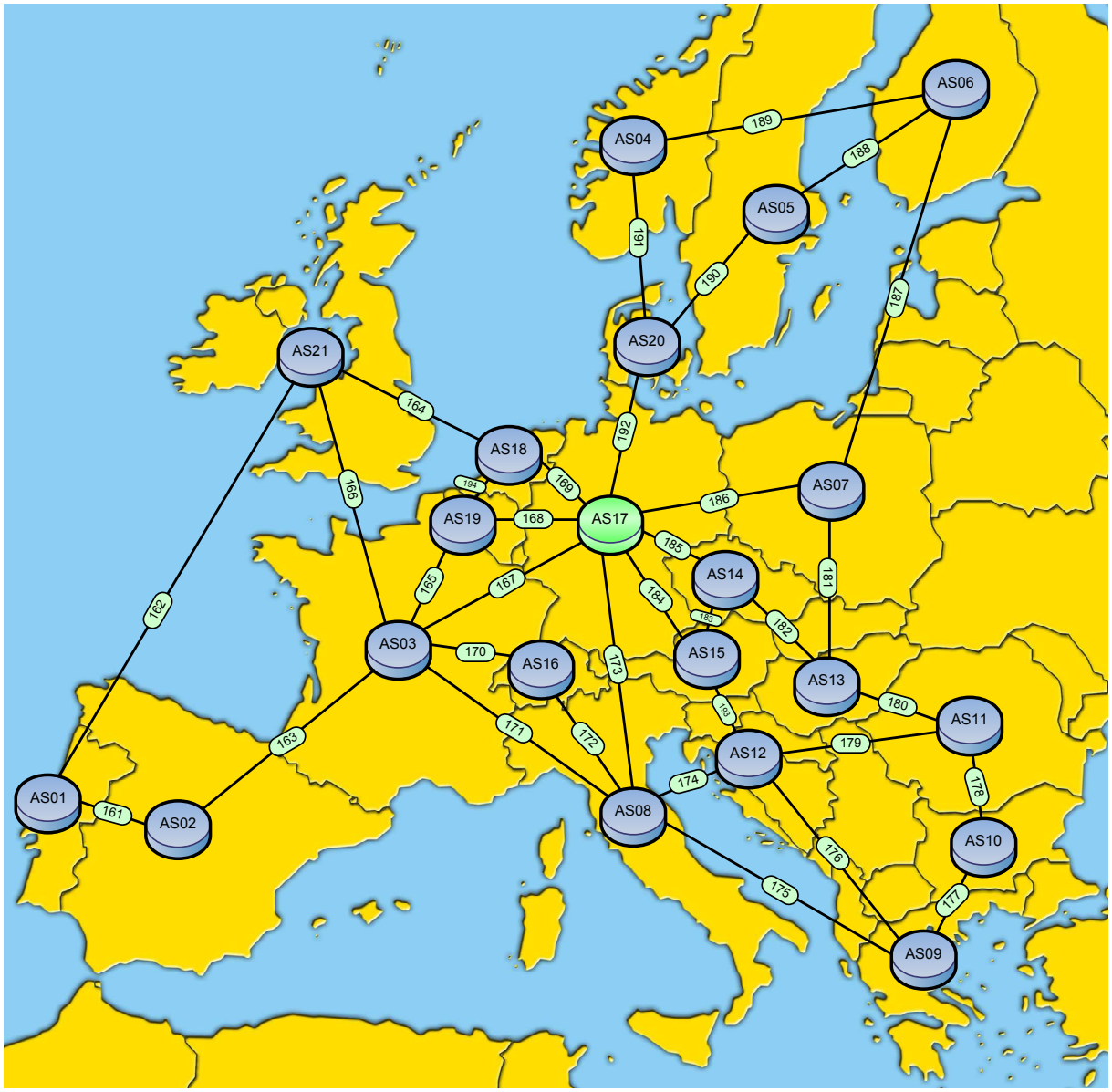


Figura E.1: Escenario 5: Topología de red con direcciones.

En primer lugar se muestran las tablas obtenidas utilizando BGP normal en las que se pueden comprobar las rutas seleccionadas para cada *peer*.

Destino	Siguiente salto	Ruta	Princ.
101.0.0.0	163.0.0.2	02 - 03 - 17	*
	161.0.0.1	21 - 03 - 17	

Tabla E.1: Escenario 5: *Adj-RIB-In* del AS01 con BGP normal.

Destino	Siguiente salto	Ruta	Princ.
101.0.0.0	163.0.0.2	03 - 17	*

Tabla E.2: Escenario 5: *Adj-RIB-In* del AS02 con BGP normal.

Destino	Siguiente salto	Ruta	Princ.
101.0.0.0	167.0.0.2	17	*
	171.0.0.2	08 - 17	
	165.0.0.2	19 - 17	

Tabla E.3: Escenario 5: *Adj-RIB-In* del AS03 con BGP normal.

Destino	Siguiente salto	Ruta	Princ.
101.0.0.0	191.0.0.2	20 - 17	*
	189.0.0.2	06 - 07 - 17	

Tabla E.4: Escenario 5: *Adj-RIB-In* del AS04 con BGP normal.

Destino	Siguiente salto	Ruta	Princ.
101.0.0.0	190.0.0.2	20 - 17	*
	188.0.0.2	06 - 07 - 17	

Tabla E.5: Escenario 5: *Adj-RIB-In* del AS05 con BGP normal.

Destino	Siguiente salto	Ruta	Princ.
101.0.0.0	187.0.0.2	07 - 17	*
	189.0.0.1	04 - 20 - 17	
	188.0.0.1	05 - 20 - 17	

Tabla E.6: Escenario 5: *Adj-RIB-In* del AS06 con BGP normal.

Destino	Siguiente salto	Ruta	Princ.
101.0.0.0	186.0.0.2	17	*

Tabla E.7: Escenario 5: *Adj-RIB-In* del AS07 con BGP normal.

Destino	Siguiente salto	Ruta	Princ.
101.0.0.0	173.0.0.2	17	*
	171.0.0.1	03 - 17	
	172.0.0.2	16 - 03 - 17	

Tabla E.8: Escenario 5: *Adj-RIB-In* del AS08 con BGP normal.

Destino	Siguiente salto	Ruta	Princ.
101.0.0.0	175.0.0.1	08 - 17	*
	176.0.0.2	12 - 08 - 17	

Tabla E.9: Escenario 5: *Adj-RIB-In* del AS09 con BGP normal.

Destino	Siguiente salto	Ruta	Princ.
101.0.0.0	177.0.0.1	09 - 08 - 17	*
	178.0.0.2	11 - 13 - 07 - 17	

Tabla E.10: Escenario 5: *Adj-RIB-In* del AS10 con BGP normal.

Destino	Siguiente salto	Ruta	Princ.
101.0.0.0	180.0.0.2	13 - 07 - 17	*
	179.0.0.2	12 - 08 - 17	
	178.0.0.1	10 - 09 - 08 - 17	

Tabla E.11: Escenario 5: *Adj-RIB-In* del AS11 con BGP normal.

Destino	Siguiente salto	Ruta	Princ.
101.0.0.0	174.0.0.1	08 - 17	*
	193.0.0.2	15 - 17	
	176.0.0.1	09 - 08 - 17	
	179.0.0.1	11 - 13 - 07 - 17	

Tabla E.12: Escenario 5: *Adj-RIB-In* del AS12 con BGP normal.

Destino	Siguiente salto	Ruta	Princ.
101.0.0.0	181.0.0.1	07 - 17	*
	182.0.0.2	14 - 17	

Tabla E.13: Escenario 5: *Adj-RIB-In* del AS13 con BGP normal.

Destino	Siguiente salto	Ruta	Princ.
101.0.0.0	185.0.0.2	17	*
	183.0.0.2	15 - 17	
	182.0.0.1	13 - 07 - 17	

Tabla E.14: Escenario 5: *Adj-RIB-In* del AS14 con BGP normal.

Destino	Siguiente salto	Ruta	Princ.
101.0.0.0	184.0.0.2	17	*
	183.0.0.1	14 - 17	
	193.0.0.1	12 - 08 - 17	

Tabla E.15: Escenario 5: *Adj-RIB-In* del AS15 con BGP normal.

Destino	Siguiente salto	Ruta	Princ.
101.0.0.0	170.0.0.1	03 - 17	*
	172.0.0.1	08 - 17	

Tabla E.16: Escenario 5: *Adj-RIB-In* del AS16 con BGP normal.

Destino	Siguiente salto	Ruta	Princ.
101.0.0.0	169.0.0.1	17	*
	194.0.0.2	19 - 17	
	164.0.0.2	21 - 03 - 17	

Tabla E.17: Escenario 5: *Adj-RIB-In* del AS18 con BGP normal.

Destino	Siguiente salto	Ruta	Princ.
101.0.0.0	168.0.0.1	17	*
	165.0.0.1	03 - 17	
	194.0.0.1	18 - 17	

Tabla E.18: Escenario 5: *Adj-RIB-In* del AS19 con BGP normal.

Destino	Siguiente salto	Ruta	Princ.
101.0.0.0	192.0.0.1	17	*

Tabla E.19: Escenario 5: *Adj-RIB-In* del AS20 con BGP normal.

Destino	Siguiente salto	Ruta	Princ.
101.0.0.0	166.0.0.1	03 - 17	*
	164.0.0.1	18 - 17	
	162.0.0.1	01 - 02 - 03 - 17	

Tabla E.20: Escenario 5: *Adj-RIB-In* del AS21 con BGP normal.

En las tablas anteriores se puede comprobar como todos y cada uno de los *peers* de la red han conseguido una ruta para alcanzar el destino que está en AS17. Como sucedía en el resto de escenarios, no se muestra la tabla de prefijos del *peer* AS17 por tratarse del Sistema Autónomo en el que se encuentra la red destino.

Con la figura E.2 se pueden verificar las rutas obtenidas por cada *peer* y con ella se tiene una mejor visualización de los resultados obtenidos para la utilización de BGP normal..

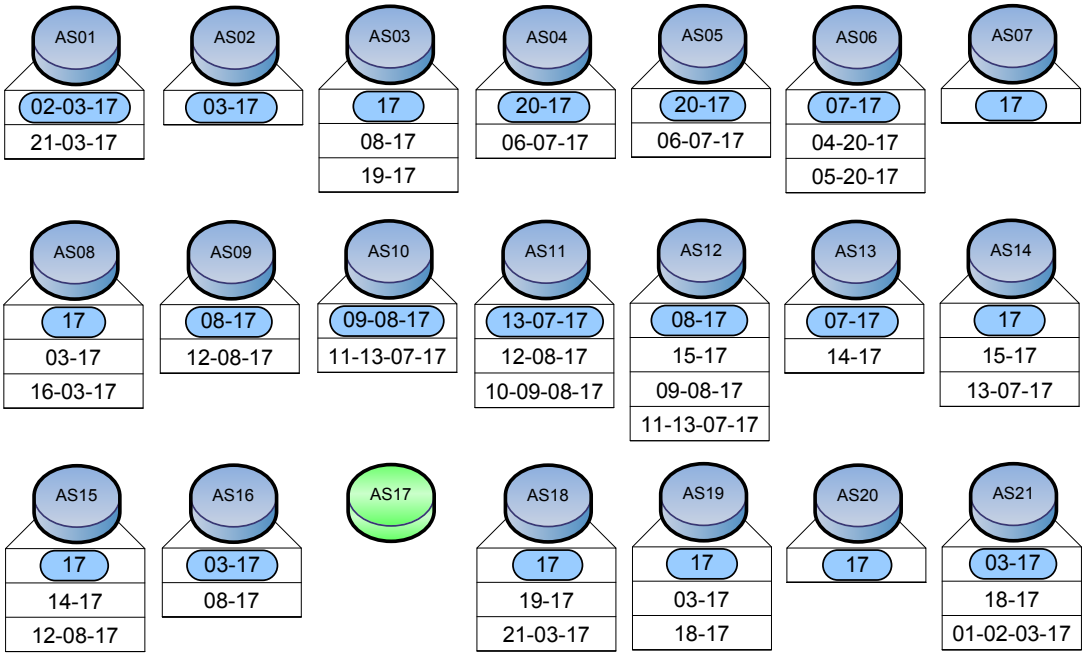


Figura E.2: Escenario 5: RIB´s obtenidas con BGP normal.

A continuación se mostrarán las tablas de rutas de cada uno de los *peers* para el caso de utilizar la implementación del protocolo BGP desarrollada en este proyecto.

Destino	Tipo de ruta	Siguiente salto	Ruta	Princ.	Secund.
101.0.0.0	P	161.0.0.2	02 - 03 - 17	*	
	S	162.0.0.2	21 - 18 - 17		*
	P	162.0.0.2	21 - 03 - 17		

Tabla E.21: Escenario 5: *Adj-RIB-In* del AS01 con BGP modificado.

Destino	Tipo de ruta	Siguiente salto	Ruta	Princ.	Secund.
101.0.0.0	P	163.0.0.2	03 - 17	*	
	S	161.0.0.2	01 - 21 - 18 - 17		*

Tabla E.22: Escenario 5: *Adj-RIB-In* del AS02 con BGP modificado.

Destino	Tipo de ruta	Siguiente salto	Ruta	Princ.	Secund.
101.0.0.0	P	167.0.0.2	17	*	
	P	165.0.0.2	19 - 17		*
	P	171.0.0.2	08 - 17		
	S	170.0.0.2	16 - 08 - 17		
	S	166.0.0.2	21 - 08 - 17		
	S	163.0.0.1	02 - 01 - 21 - 18 - 17		

Tabla E.23: Escenario 5: *Adj-RIB-In* del AS03 con BGP modificado.

Destino	Tipo de ruta	Siguiente salto	Ruta	Princ.	Secund.
101.0.0.0	P	191.0.0.2	20 - 17	*	
	P	189.0.0.2	06 - 07 - 17		*

Tabla E.24: Escenario 5: *Adj-RIB-In* del AS04 con BGP modificado.

Destino	Tipo de ruta	Siguiente salto	Ruta	Princ.	Secund.
101.0.0.0	P	190.0.0.2	20 - 17	*	
	P	188.0.0.2	06 - 07 - 17		*
	S	188.0.0.2	06 - 04 - 20 - 17		
	S	190.0.0.2	20 - 04 - 06 - 07 - 17		

Tabla E.25: Escenario 5: *Adj-RIB-In* del AS05 con BGP modificado.

Destino	Tipo de ruta	Siguiente salto	Ruta	Princ.	Secund.
101.0.0.0	P	187.0.0.2	07 - 17	*	
	P	189.0.0.1	04 - 20 - 17		*
	P	188.0.0.1	05 - 20 - 17		
	S	187.0.0.2	07 - 13 - 14 - 17		

Tabla E.26: Escenario 5: *Adj-RIB-In* del AS06 con BGP modificado.

Destino	Tipo de ruta	Siguiente salto	Ruta	Princ.	Secund.
101.0.0.0	P	186.0.0.2	17	*	
	S	181.0.0.2	13 - 14 - 17		*
	S	187.0.0.1	06 - 04 - 20 - 17		

Tabla E.27: Escenario 5: *Adj-RIB-In* del AS07 con BGP modificado.

Destino	Tipo de ruta	Siguiente salto	Ruta	Princ.	Secund.
101.0.0.0	P	173.0.0.2	17	*	
	P	171.0.0.1	03 - 17		*
	S	171.0.0.1	03 - 19 - 17		
	P	172.0.0.2	16 - 03 - 17		
	P	174.0.0.2	12 - 15 - 17		
	S	175.0.0.2	09 - 12 - 15 - 17		

Tabla E.28: Escenario 5: *Adj-RIB-In* del AS08 con BGP modificado.

Destino	Tipo de ruta	Siguiente salto	Ruta	Princ.	Secund.
101.0.0.0	P	175.0.0.1	08 - 17	*	
	P	176.0.0.2	12 - 15 - 17		*
	S	175.0.0.1	08 - 03 - 17		
	S	176.0.0.2	12 - 08 - 17		
	S	177.0.0.2	10 - 11 - 12 - 15 - 17		

Tabla E.29: Escenario 5: *Adj-RIB-In* del AS09 con BGP modificado.

Destino	Tipo de ruta	Siguiente salto	Ruta	Princ.	Secund.
101.0.0.0	P	177.0.0.1	09 - 08 - 17	*	
	P	178.0.0.2	11 - 12 - 15 - 17		*
	S	178.0.0.2	11 - 13 - 07 - 17		
	S	177.0.0.1	09 - 12 - 15 - 17		

Tabla E.30: Escenario 5: *Adj-RIB-In* del AS10 con BGP modificado.

Destino	Tipo de ruta	Siguiente salto	Ruta	Princ.	Secund.
101.0.0.0	P	179.0.0.2	12 - 15 - 17	*	
	P	180.0.0.2	13 - 07 - 17		*
	S	179.0.0.2	12 - 08 - 17		
	S	180.0.0.2	13 - 14 - 17		
	P	178.0.0.1	10 - 09 - 08 - 17		

Tabla E.31: Escenario 5: *Adj-RIB-In* del AS11 con BGP modificado.

Destino	Tipo de ruta	Siguiente salto	Ruta	Princ.	Secund.
101.0.0.0	P	193.0.0.2	15 - 17	*	
	P	174.0.0.1	08 - 17		*
	S	174.0.0.1	08 - 03 - 17		
	P	176.0.0.1	09 - 08 - 17		
	P	179.0.0.1	11 - 13 - 07 - 17		

Tabla E.32: Escenario 5: *Adj-RIB-In* del AS12 con BGP modificado.

Destino	Tipo de ruta	Siguiente salto	Ruta	Princ.	Secund.
101.0.0.0	P	181.0.0.1	07 - 17	*	
	P	182.0.0.2	14 - 17		*
	S	182.0.0.2	14 - 15 - 17		
	P	180.0.0.1	11 - 12 - 15 - 17		

Tabla E.33: Escenario 5: *Adj-RIB-In* del AS13 con BGP modificado.

Destino	Tipo de ruta	Siguiente salto	Ruta	Princ.	Secund.
101.0.0.0	P	185.0.0.2	17	*	
	P	183.0.0.2	15 - 17		*
	P	182.0.0.1	13 - 07 - 17		

Tabla E.34: Escenario 5: *Adj-RIB-In* del AS14 con BGP modificado.

Destino	Tipo de ruta	Siguiente salto	Ruta	Princ.	Secund.
101.0.0.0	P	184.0.0.2	17	*	
	P	183.0.0.1	14 - 17		*
	P	193.0.0.1	12 - 08 - 17		

Tabla E.35: Escenario 5: *Adj-RIB-In* del AS15 con BGP modificado.

Destino	Tipo de ruta	Siguiente salto	Ruta	Princ.	Secund.
101.0.0.0	P	170.0.0.1	03 - 17	*	
	P	172.0.0.1	08 - 17		*
	S	172.0.0.1	08 - 03 - 17		
	S	170.0.0.1	03 - 19 - 17		

Tabla E.36: Escenario 5: *Adj-RIB-In* del AS16 con BGP modificado.

Destino	Tipo de ruta	Siguiente salto	Ruta	Princ.	Secund.
101.0.0.0	P	169.0.0.1	17	*	
	P	194.0.0.2	19 - 17		*
	P	164.0.0.2	21 - 03 - 17		
	S	194.0.0.2	19 - 03 - 17		

Tabla E.37: Escenario 5: *Adj-RIB-In* del AS18 con BGP modificado.

Destino	Tipo de ruta	Siguiente salto	Ruta	Princ.	Secund.
101.0.0.0	P	168.0.0.1	17	*	
	P	165.0.0.1	03 - 17		*
	P	194.0.0.1	18 - 17		

Tabla E.38: Escenario 5: *Adj-RIB-In* del AS19 con BGP modificado.

Destino	Tipo de ruta	Siguiente salto	Ruta	Princ.	Secund.
101.0.0.0	P	192.0.0.1	17	*	
	S	191.0.0.1	04 - 06 - 07 - 17		*
	S	190.0.0.1	05 - 06 - 07 - 17		

Tabla E.39: Escenario 5: *Adj-RIB-In* del AS20 con BGP modificado.

Destino	Tipo de ruta	Siguiente salto	Ruta	Princ.	Secund.
101.0.0.0	P	166.0.0.1	03 - 17	*	
	P	164.0.0.1	18 - 17		*
	P	162.0.0.1	01 - 02 - 03 - 17		

Tabla E.40: Escenario 5: *Adj-RIB-In* del AS21 con BGP modificado.

Con la figura E.3 se puede verificar la obtención de dos rutas, principal y secundaria disjunta, por cada *peer* de la red. Esta figura también permite una mejor visualización de los resultados obtenidos para la utilización del protocolo BGP modificado en este proyecto.

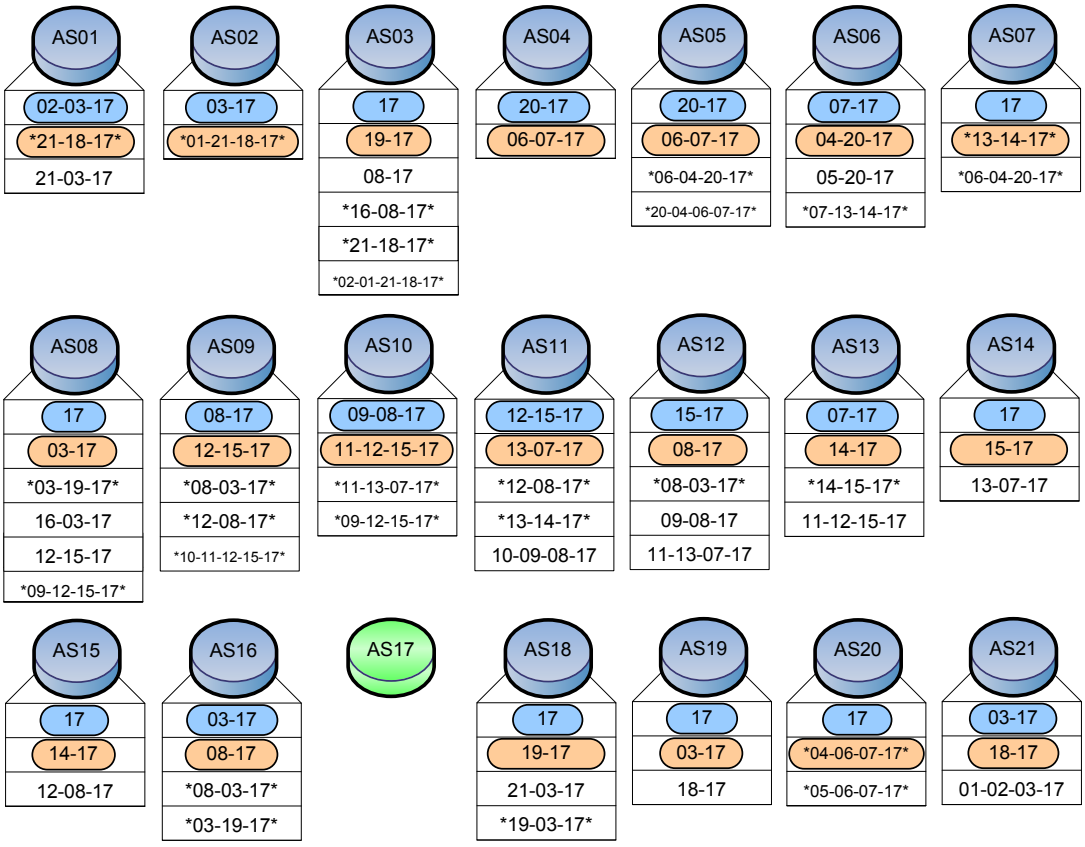


Figura E.3: Escenario 5: RIB´s obtenidas con BGP modificado.

Bibliografía

[1] Ricardo Romeral Ortega. *Mecanismos de protección en escenarios IP-MPLS multidominio.* Leganés (España), Julio 2007.

[2] Changcheng Huang and Donald Messier. *A fast and scalable inter-domain MPLS protection mechanism.* Marzo 2004.

[3] Y. Rekhter J.P. Lang and D. Papadimitriou. *RSVP-TE Extensions in support of End-to-End Generalized Multi-Protocol Label Switching (GMPLS)-based Recovery.* Octubre 2006.

[4] VNUML Web Server: http://www.dit.upm.es/vnumlwiki/index.php/Main_Page.

[5] Kunihiro Ishiguro et al. *Quagga: a routing software package for TCP/IP networks.* Junio 2005.

[6] T.J. Watson Research Center Y. Rekhter and T. Li. A border gateway protocol 4 (bgp-4). RFC 1771, Marzo 1995.

[7] Iljitsch Van Beijnum. *Building Reliable Networks with the Border Gateway Protocol.* O´Reilly (EE.UU.), 2002.

[8] BGP Wikipedia: http://es.wikipedia.org/wiki/Border_Gateway_Protocol.

[9] Quagga Routing Suite: http://www.uk.quagga.net.

[10] Byron S. Gottfried. *Programación en c.* McGraw-Hill, 2005.

[11] Euro 6IX - IPv6: The New Internet: http://www.euro6ix.org.

[12] Source Forge: http://sourceforge.net/projects/vnuml/.

[13] Fermin Galán Márquez. *Tecnología XML en la herramienta de simulación de redes VNUML.* Marzo 2004.

[14] Wireshark: http://www.wireshark.org.